



ugr

Universidad  
de Granada

TRABAJO FIN DE GRADO  
GRADO EN INGENIERIA INFORMATICA

# App diario de ejercicio físico

---

**Autor**

Francisco de Asís Carrasco Conde

**Directora**

María José Rodríguez Fórtiz



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Granada, Junio de 2025

# **App diario de ejercicio físico**

Francisco de Asís Carrasco Conde

## **Resumen**

Este TFG tratará sobre la investigación de tecnologías y algunas apps del mercado relacionadas con el gremio, y el desarrollo de una app para dispositivos móviles para guardar y visualizar datos relacionados con el ejercicio físico.

---

Yo, **Francisco de Asís Carrasco Conde**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 26301846N, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco de Asís Carrasco Conde

Granada a X de Junio de 2025.

---

D. **María José Rodríguez Fórtiz**, Profesora del Departamento de Lenguajes y Sistemas Informáticos

**Informo:**

Que el presente trabajo, titulado ***EjercitaTec***, ha sido realizado bajo mi supervisión por **Francisco de Asís Carrasco Conde**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2025.

**El/la director(a)/es:**

**María José Rodríguez Fórtiz**

# Índice general

<b>1. Introducción</b>	<b>11</b>
1.1. Justificación . . . . .	11
1.2. Objetivos . . . . .	12
1.3. Estructura de la Memoria . . . . .	13
<b>2. Estado del arte</b>	<b>15</b>
2.1. Accesibilidad . . . . .	15
2.2. Dominio del problema . . . . .	16
2.3. Aplicaciones similares . . . . .	17
2.4. Metodologías potenciales a aplicar . . . . .	23
2.5. Tecnologías potenciales para usar . . . . .	23
2.5.1. Base de datos . . . . .	24
2.5.2. Backend . . . . .	25
2.5.3. Frontend . . . . .	27
<b>3. Propuesta</b>	<b>29</b>
3.1. Descripción detallada . . . . .	29
3.2. Elección de tecnologías . . . . .	30
3.3. Backend . . . . .	30
3.4. Frontend . . . . .	30
3.5. Base de datos . . . . .	30
3.6. Diagrama de arquitectura . . . . .	31
3.7. Metodología utilizada . . . . .	32
3.8. Temporización . . . . .	32
3.9. Desarrollo . . . . .	32

3.9.1. Iteración 0 . . . . .	33
3.9.2. Iteración 1 . . . . .	60
3.9.3. Iteración 2 . . . . .	69
3.9.4. Iteración 3 . . . . .	80
3.9.5. Iteración 4 . . . . .	91
3.9.6. Iteración 5 . . . . .	101
<b>4. Conclusiones</b>	<b>103</b>
4.1. Objetivos . . . . .	103
4.2. Valoración personal . . . . .	104
4.3. Objetivos de desarrollo sostenible . . . . .	105
4.4. Futuro empresarial . . . . .	105
4.5. Trabajos futuros . . . . .	106

## Índice de figuras

2.1. Fitbit . . . . .	18
2.2. Apple fitness . . . . .	19
2.3. Strava . . . . .	20
2.4. Pulsera whoop . . . . .	21
2.5. Tabla comparativa de aplicaciones similares . . . . .	22
2.6. Tabla comparativa tipos de bases de datos . . . . .	25
2.7. Tabla comparativa frameworks para el backend . . . . .	27
2.8. Tabla comparativa frameworks para el frontend . . . . .	28
3.1. Diagrama de arquitectura . . . . .	31
3.2. Pagina inicial . . . . .	35
3.3. Inicio de sesion . . . . .	35
3.4. Crear cuenta 1 . . . . .	36
3.5. Crear cuenta 2 . . . . .	36
3.6. Menu principal . . . . .	37
3.7. Lista ejercicios . . . . .	38
3.8. Lista ejercicios filtrada . . . . .	38
3.9. Datos ejercicio . . . . .	39
3.10. Pop up graficar segun tiempo . . . . .	39
3.11. Pop up establecer meta . . . . .	39
3.12. Pop up crear ejercicio . . . . .	39
3.13. Lista rutinas . . . . .	40
3.14. Lista rutinas filtradas . . . . .	40
3.15. Datos rutina modificable . . . . .	40
3.16. Datos rutina no modificable . . . . .	40
3.17. Modificar ejercicios rutina . . . . .	41
3.18. Cambiar orden ejercicios rutina . . . . .	41
3.19. Ejercicios rutina descargada . . . . .	42

3.20. Pop up crear rutinas . . . . .	42
3.21. Opciones de perfil usuario . . . . .	43
3.22. Pop up de confirmacion . . . . .	43
3.23. Pop up de confirmacion resumir datos . . . . .	43
3.24. Buscar usuario . . . . .	44
3.25. Buscar usuario filtrado . . . . .	44
3.26. Rutinas de un usuario . . . . .	45
3.27. Widget descargar rutina . . . . .	45
3.28. Widget rutina descargada . . . . .	46
3.29. Buscar rutinas por su nombre filtrada . . . . .	46
3.30. Entrenamiento de un día determinado . . . . .	47
3.31. Descanso en un día determinado . . . . .	47
3.32. Frame 29.png . . . . .	48
3.33. Lista ejercicios del entrenamiento actual . . . . .	49
3.34. Primera serie flexiones . . . . .	49
3.35. Realizando flexiones . . . . .	50
3.36. Fin de la serie(Descanso no completado) . . . . .	50
3.37. Fin de la serie(Descanso completado) . . . . .	51
3.38. Acabar ejercicio o añadir serie . . . . .	51
3.39. Realizando ejercicio midiendo tiempo . . . . .	52
3.40. Fin entrenamiento . . . . .	52
3.41. Metas cumplidas . . . . .	53
3.42. Datos entrenamiento terminado . . . . .	53
3.43. Chat con IA . . . . .	54
3.44. Datos detallados entrenamiento terminado . . . . .	55
3.45. Pop up datos de un ejercicio en entrenamiento . . . . .	55
3.46. Backlog prioridades MoSCoW . . . . .	57
3.47. Backlog prioridades MoSCoW . . . . .	58
3.48. Iteracion 1 . . . . .	59
3.49. Iteracion 2 . . . . .	59
3.50. Iteracion 3 . . . . .	59
3.51. Iteracion 4 . . . . .	60
3.52. Iteracion 5 . . . . .	60
3.53. Iteracion 6 . . . . .	60
3.54. Resultado de la implementacion de la lista de ejercicios . . . . .	61
3.55. Crear ejercicios . . . . .	63
3.56. Modificar descripcion ejercicios . . . . .	63
3.57. Pantalla donde se muestran los datos del ejercicio al acceder desde la lista . . . . .	64
3.58. Nueva meta . . . . .	65
3.59. PopUp Confirmacion . . . . .	66
3.60. BD local iteracion 1 . . . . .	67
3.61. Pantalla nueva . . . . .	68
3.62. Pantalla antigua . . . . .	68



3.63. Menu Principal semifuncional . . . . .	70
3.64. Log In / Sing In . . . . .	71
3.65. Sing In pantalla 1 . . . . .	71
3.66. Sing In pantalla 2 . . . . .	72
3.67. Log In / Sing In . . . . .	73
3.68. Sing In pantalla 1 . . . . .	73
3.69. Lista Rutinas . . . . .	75
3.70. Datos Rutina . . . . .	76
3.71. Crear Rutina . . . . .	76
3.72. Lista Ejercicios Rutina . . . . .	77
3.73. Modificar Rutina . . . . .	77
3.74. Lista añadir ejercicio a rutina . . . . .	78
3.75. BD local iteracion 2 . . . . .	79
3.76. BD servidor iteracion 2 . . . . .	80
3.77. Peso Objetivo . . . . .	81
3.78. Elegir lista compartida por el usuario o locales . . . . .	83
3.79. Rutinas compartidas por el usuario . . . . .	83
3.80. Datos rutinas compartidas por el usuario . . . . .	84
3.81. Buscar rutinas o usuario . . . . .	86
3.82. Buscar rutinas por su nombre . . . . .	86
3.83. Buscar usuarios . . . . .	87
3.84. Datos rutinas para descargar . . . . .	88
3.85. BD del servidor iteracion 3 . . . . .	90
3.86. Calendario . . . . .	93
3.87. Calendario con eventos . . . . .	93
3.88. Descanso . . . . .	93
3.89. Entrenar . . . . .	93
3.90. Lista inicial de ejercicios . . . . .	95
3.91. Realizando serie . . . . .	95
3.92. Guardando marcas . . . . .	95
3.93. Pantalla enter series . . . . .	95
3.94. Flujo entrenamiento . . . . .	96
3.95. Meta superada . . . . .	97
3.96. Registrar peso por día . . . . .	98
3.97. BD local iteración 4 . . . . .	100
3.98. BD del servidor iteración 4 . . . . .	101



# Capítulo 1

## Introducción

### 1.1. Justificación

En este TFG se va a desarrollar una aplicación que ayude a las personas a realizar ejercicio físico de forma controlada y a monitorizar las metas que se van alcanzando. Primero de todo, ¿qué lleva a hacer este TFG? Para empezar, es verdad que hay muchas apps en los repositorios que están pensadas para ser usadas por usuarios de gimnasio, para hacer más fácil el seguimiento y evolución. Sin embargo, la gente que suele hacer deporte de manera más informal o por hobby no suele tener los conocimientos o dispositivos para hacer mediciones de calidad e interpretarlas correctamente, por lo que necesitarían una aplicación más sencilla tipo agenda para planificar y monitorizar sus ejercicios.

Otra problemática que encontramos en las apps existentes suele ser la falta de accesibilidad, pensando en usuarios con algún tipo de discapacidad que quieran realizar ejercicios. Por ejemplo, encontramos que los scrolls abundan, hay eventos no controlados por el usuario, colores confusos para daltónicos, a veces hay ausencias de iconos asociados a botones y listas para facilitar su comprensión, y un amplio etcétera.

Otras carencias que se observan es que no suelen incluir la funcionalidad de ver entrenamientos recomendados por otros. La gente suele buscarlos en redes sociales o videos que se encuentran en la red, y muchas veces en estos videos se ponen a dar rodeos para rascar "más tiempo, y así es como el usuario pierde tiempo para que a lo mejor no sea el entrenamiento que el andaba buscando. Lo ideal sería que los ejercicios estuvieran bien organizados en rutinas y que estas pudieran compartirse en la aplicación.

Me veo en la necesidad de hacer esta app para dotar de una herramienta sim-

plificada, fácil de manejar, y que permita compartir información entre usuarios de forma eficaz y accesible. Simplificada porque la información que se maneja se tratará de una forma fácil e intuitiva. Fácil de manejar, dado que el usuario solo tendrá que, por ejemplo, anotar el número de veces que levanta una pesa y anotarlo en la app, con lo cuál se podrá facilitar la monitorización. Compartir información eficazmente, porque en una sección de la app habrá una parte formato red social, que permitirá tanto buscar usuarios que comparten entrenamientos, como los propios entrenamientos en si, acompañados de su descripción en la que el creador da una breve información acerca del entrenamiento. Accesible, porque se seguirán guías de diseño para facilitar el uso por usuarios de diversas capacidades.

Una vez explicado esto, ¿cómo se le dará soporte a los usuarios? Creando una app que le ayude de la siguiente forma: Ayudándole a planificar o usar rutinas de entrenamientos compuestas por series de ejercicios; Facilitándole la monitorización y supervisión de su realización, al poder introducir metas a alcanzar en parámetros, que pueden ser repeticiones/peso/distancia/tiempo, y que se pueden medir de forma independiente o al mismo tiempo. También guardando la cantidad de series de un ejercicio realizadas en un entrenamiento con sus respectivos parámetros, y contabilizándole al usuario el tiempo descansado, para facilitar su correcto entrenamiento. Además, si el usuario lo solicita se le enseñará un resumen de su rendimiento en cada ejercicio con respecto a la fecha en la que se realizó el mismo, para que vea su evolución respecto al tiempo.

También la app ayudará a hacer un control de las metas que se proponga el usuario, es decir, si el usuario se propone bajar de peso o aumentar su rendimiento en un ejercicio o deporte concreto, la app le recordará las metas que se autoproponga y le avisará cuando las cumpla.

## 1.2. Objetivos

Nuestro objetivo general es el desarrollo de una aplicación móvil multiplataforma para la planificación y monitorización de ejercicio físico en la que se puedan marcar metas personales respecto a los deportes o ejercicios que desee el usuario.

Los objetivos específicos de este trabajo de fin de grado son:

- Analizar algunas apps del mercado, así como sus características, qué ofrecen, su costo para el usuario y las valoraciones de los usuarios finales, para aclarar qué es lo que buscan los usuarios en este tipo de apps y considerar esas necesidades en el software a desarrollar.
- Investigar sobre qué herramientas usar para la implementación de la base de datos, backend, frontend, y técnicas y metodologías para dar un desarrollo de calidad al software y para ayudar a la sostenibilidad del sistema en el

tiempo.

- Conocer y aprender a usar herramientas actuales y punteras para el desarrollo de aplicaciones móviles.
- Desarrollar una aplicación multiplataforma que cubra todos los requisitos necesarios para dar soporte a la planificación y realización de entrenamientos y monitorización de rutinas de ejercicio físico.
- Tratar de obtener un software lo más accesible posible, siguiendo las guías de diseño estandarizadas.

### 1.3. Estructura de la Memoria

El 1º capítulo(**Introducción**) es una descripción sobre lo que se va a abordar y qué ideas iniciales se tienen acerca del software que se va a desarrollar y algunos aspectos importantes del gremio en el que se usaría.

El 2º capítulo(**Estado del arte**), es un análisis de cómo está el ámbito sobre el que se va a desarrollar este trabajo, es decir, apps del mercado(sus prestaciones y funciones más importantes) y frameworks que se podrían emplear para el desarrollo, así como sus ventajas y desventajas.

El 3º capítulo (**Propuesta**), es la explicación de las decisiones tomadas durante el desarrollo y de como se ha desarrollado el mismo.

El 4º capítulo(**Conclusiones y trabajos futuros**), una reflexión final sobre el trabajo realizado.



## Capítulo 2

# Estado del arte

## 2.1. Accesibilidad

La "Guía de accesibilidad de aplicaciones móviles." ofrece pautas esenciales para el desarrollo y evaluación de apps accesibles, dirigidas a personas con discapacidades. La guía se basa en la Directiva (UE) 2016/2102 y el Real Decreto 1112/2018, los cuales exigen la accesibilidad de aplicaciones en el sector público a partir de 2021.

Algunos puntos clave incluyen:

1. Dificultades que enfrentan personas con discapacidades sensoriales, motrices y cognitivas, destacando la necesidad de adaptar las aplicaciones para estos perfiles de usuarios.
2. Principios de desarrollo accesible, enfatiza la importancia de integrar la accesibilidad desde las primeras etapas del diseño y desarrollo, usando las herramientas de accesibilidad de los sistemas operativos.
3. Se detallan directrices sobre cómo hacer que las interfaces de usuario sean accesibles, incluyendo la configuración del contraste, la visibilidad del foco y el tamaño de los componentes interactivos.
4. La guía ofrece pautas para validar la accesibilidad de las aplicaciones mediante herramientas automáticas y manuales.
5. Proporciona una lista de herramientas y productos de apoyo para facilitar la interacción de personas con discapacidades, como lectores de pantalla y magnificadores.

En resumen, la guía ayuda a los desarrolladores a crear aplicaciones móviles que sean accesibles a todos los usuarios, mejorando la inclusión social y cumpliendo con los requisitos legales de accesibilidad.

## 2.2. Dominio del problema

Para comenzar, es necesario hablar un poco sobre conceptos generales en el mundo del deporte, definiendo algunos conceptos.

Un **ejercicio** es la repetición de un movimiento varias veces para estimular uno o varios músculos. Las **rutinas**, entendámoslos como la colección de distintos ejercicios destinados a entrenar una parte o varias del cuerpo, a las rutinas también se les llama **entrenamientos**, pero en la app se referirá a rutinas como la anterior definición y los entrenamientos serán el hecho de haber realizado una rutina. Ejemplo, el entrenamiento del día 6 es hacer la rutina de flexiones. Una **serie** es cuando dentro de un ejercicio repetimos ese movimiento un número determinado de veces. Entre series se hacen descansos. Una **repetición**, es la realización del movimiento de ese ejercicio en una serie, es decir, si por ejemplo se están haciendo flexiones, se hacen 12, descanso, se hacen 10, descanso, se hacen 9 y ya no se hacen más flexiones, dentro del entrenamiento se vería así:

### Rutina de entrenamiento 1:

- Ejercicio 1
  - Serie 1: X repeticiones
  - Descanso
  - Serie 2: X repeticiones
- Flexiones
  - Serie 1: 12 repeticiones
  - Descanso
  - Serie 2: 10 repeticiones
  - Descanso
  - Serie 3: 9 repeticiones
- Ejercicio 3
  - Serie 1: X repeticiones
  - Descanso
  - Serie 2: X repeticiones

Las metas son objetivos que se pone el usuario en un ejercicio, por ejemplo, si un usuario desea llegar a hacer 10 flexiones mínimo en una serie la proxima vez que entrene, es que se ha puesto una meta de 10 flexiones.

El compartir una rutina es publicar los detalles de una rutina para que otra persona la pueda hacer en sus entrenamientos.

En este gremio del deporte es muy importante la planificación, ya que si se entrena muy de seguido un músculo se pueden producir lesiones o hacer que los entrenamientos sean inútiles, lo que se conoce como sobrentrenamiento. Por su contraparte, es importante no dejar de entrenar todas las zonas del cuerpo, ya que se puede lastrar una musculatura debil para otros tipos de entrenamientos. También es importante planificar los días de descansos, para evitar entrenamientos con el cuerpo fatigado.



Ya dentro de los entrenamientos también es importante respetar los descansos. Dado que si se descansa menos tiempo del debido puede provocar lesiones de muchos tipos, incluidos neuronales (Leonard et al., 2018) dado al estrés que sufre el cerebro durante el deporte.

Aunque se sigan correctamente las pautas mencionadas en los 2 párrafos anteriores, puede que el deportista no vea los frutos de sus entrenamientos porque o bien no se acuerda de su rendimiento de hace 1 mes y no sabe si es el mismo o no, o simplemente es un sentimiento placebo. Por ello sería útil tener un historial de entrenamiento de fácil acceso o incluso gráficas, para ver si el deportista mejora su rendimiento o no.

## 2.3. Aplicaciones similares

En esta sección describiremos algunas de las aplicaciones que existen similares a la propuesta. Las que permiten mediciones de constantes relacionadas durante el entrenamiento físico suelen ser de pago.

Vamos a nombrar algunos ejemplos de aplicaciones y describirlas brevemente, incluyendo su precio. Pondremos al final una tabla comparativa de ellas:

- **Fitbit Premium: €9.99**, esta app ofrece una interfaz para medir constantes usando un smartwatch con sensor cardíaco (ritmo cardíaco, calorías quemadas, pasos, ect). En lo que respecta que es el entrenamiento, solo ofrece varios entrenamientos prefijados, además no permite guardar resultados. [Página oficial](#)

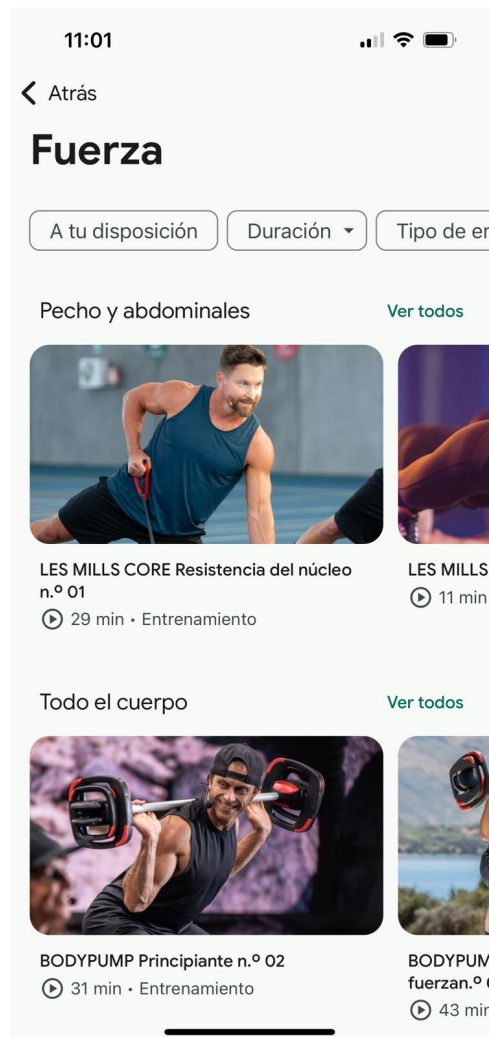


Figura 2.1: Fitbit

- **Apple Fitness+:** €9.99, en su versión gratuita nos permite guardar todas las variables permitidas con un smartwatch, las registra y va aconsejando, salta un aviso si detecta que tienes estrés, realizas poco movimiento o tienes pulsaciones anormales. En la versión de pago añade los entrenamientos, pero vuelve a la misma problemática, son entrenamientos prefijados y no permite guardar el rendimiento del usuario. [Página oficial](#)



Figura 2.2: Apple fitness

- **Strava Premium: €5.99**, es una app más enfocada a running, pero permite medir variables relacionadas con este tipo de entrenamientos, kilómetros recorridos, ritmo, ruta recorrida, tiempo, etc. Es muy completa solo para ese tipo de entrenamientos. [Página oficial](#)

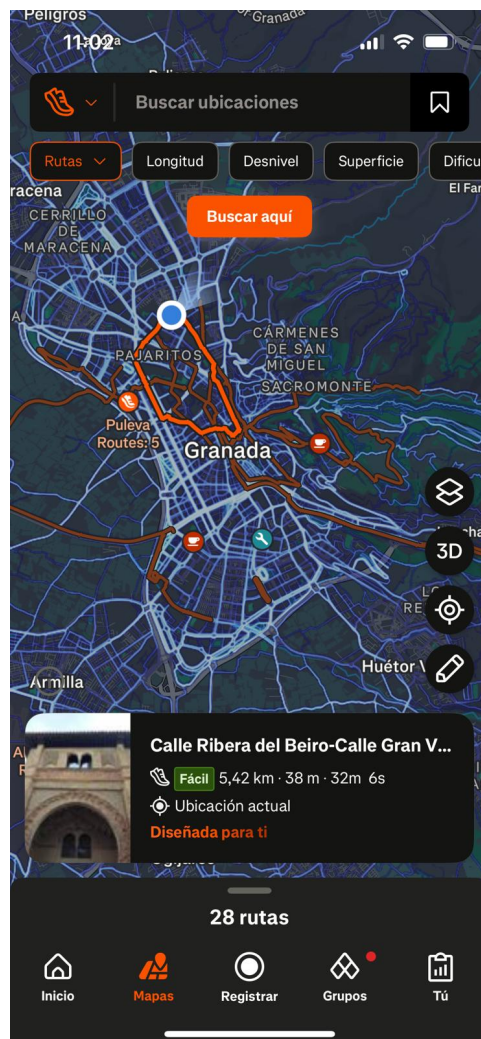


Figura 2.3: Strava

- **Whoop: €28**, de las aplicaciones esta es de lejos la más completa, permite medir estrés, cansancio del usuario, calidad del entrenamiento, calidad de sueño, recuperación y edad de tu cuerpo según tus constantes. Además, incluye una IA especializada para chatear y recibir consejos, etc. Sin embargo, es la que tiene la mensualidad más cara y solo funciona con un reloj de su marca, es decir, el usuario está obligado a comprar ese reloj si quiere usar dicha app, el reloj en concreto vale 200€ mínimo. [Página oficial](#)



Figura 2.4: Pulsera whoop

Es verdad que algunas aplicaciones tienen versión gratuita, pero no ofrecen la totalidad de sus funcionalidades, de hecho, estas versiones suelen ser extremadamente limitadas. Otras aplicaciones sí que son gratuitas, pero te obligan a usar un smartwatch de la marca, por ejemplo, la app Garmin Connect requiere comprar un reloj Garmin cuyo precio no baja de los 300€.

Para clarificar las diferencias con el resto de apps del mercado:

X	Registra datos alimenticios	Mide rendimiento de un entrenamie	Puedes hacer tus propios entrenamie	Puedes compartir entrenamientos/conten	Es necesario usar un smartwatch	Tienes una IA consultiva	Requiere pagar una subscripción	Permite controlar el peso del usuario	Puede ser usada offline	Establece metas para tus entrenamie
Strava	No	Solo de running	Solo de running	Solo running	No	No	Existe una versión gratuita limitada	No	No	Solo en los de runniing
Apple Fitness	Si	No	No	No	No	No	Existe una versión gratuita limitada	Si	Solo la parte gratuita	No
Fitbit	SI	No	No	No	No	No	No	Si	Solo la parte gratuita	No
Whoop	No	Si	No	No	Si	Si	Si	Si	Si	No
App presentada	No	Si	Si	Si	No	Si	No	Si	Casi en su plenitud	Si

Figura 2.5: Tabla comparativa de aplicaciones similares

Francisco de Asís Carrasco Conde

## 2.4. Metodologías potenciales a aplicar

Existen muchas formas de desarrollar software con sus respectivas ventajas, desventajas y forma de trabajar:

- **Cascada:** Es una forma de desarrollar de las más clásicas y lineales. Se hace una secuencia de fases de desarrollo, las cuales suelen ser diseño requisitos, diseño, implementación, pruebas, implementación y mantenimiento. Es una metodología fácil de entender y aplicar, eficaz para plazos de entrega estrictos. Sin embargo, no es flexible, y si se encuentran fallos en la planificación o alguna de las fases, no hay mucho tiempo de reacción.
- **Desarrollo Ágil:** En lugar de una secuencia rígida de fases, se promueve la flexibilidad, ya que se combina con reuniones con el cliente para obtener feedback del software funcional desarrollado entregado en dichas reuniones (no es un prototipo ya que solo es una demostración, no se le entrega para su uso durante el desarrollo), permitiendo corregir errores de planificación o en el propio software durante el desarrollo, sin perder mucho tiempo. Es necesario un buen feedback del cliente.
- **Lean Software Development:** Se basa en la idea de descartar todo aquello que no aporte valor para el cliente y quitarle importancia al desarrollo para ganar calidad y sostenibilidad en el tiempo al software. Esto se refleja en que primero se hace una investigación a fondo de todas las herramientas posibles a usar, su consecuente aprendizaje y en lo más tardío de todo esta la elección de que herramientas usar. Una vez hecha la elección, ya se puede empezar con el desarrollo.
- **V Model:** Es como el método cascada, pero antes de pasar a la siguiente fase se realizan las pruebas de la actual y no se procede hasta que estas estén válidas. Sigue teniendo las mismas desventajas que el modelo de cascada, pero es ideal para proyectos en los que es necesaria una alta calidad.
- **Modelo en espiral:** En cada fase se realiza planificación, diseño, desarrollo y evaluación de riesgos. Es muy flexible y esta analizando constantemente los riesgos.

## 2.5. Tecnologías potenciales para usar

Para hablar de las tecnologías a emplear, lo voy a separar en las 3 partes importantes en un desarrollo como el que se va a abordar: la base de datos, el backend y el frontend.

### 2.5.1. Base de datos

En este punto vamos a hablar de las ventajas y desventajas entre usar BDs relacionales y no relacionales, así como las herramientas a emplear en cada una para desarrollar estos servicios.

**Relacionales:** Las bases de datos relacionales o SQL, como bien sabemos nos permiten guardar datos de una manera bien definida y estructurada, permitiendo asegurar la integridad de la información almacenada. Son muy buena opción, si se prefiere una escalabilidad vertical, es decir, aumentar la capacidad de procesamiento del servidor que va a albergar la BD. Algunas de los SGBDS existentes en la actualidad son:

- MySQL
- MariaDB
- Oracle Database
- SQL server(Microsoft)
- PostgreSQL

**No relacionales:** También llamadas NoSQL, sirven para trabajar con estructuras de datos semidefinidas o no definidas. Esto quiere decir que no siguen un esquema rígido, lo cuál puede complicar consultas complejas, pero permiten una gran escalabilidad horizontal, permiten añadir más servidores para que operen con la misma base de datos, por tanto aumentar el número de peticiones que puede atender. Cabe decir que dentro de este tipo de bases de datos existen varios subtipos, cada uno especializado en una cualidad, al final se incluye una tabla comparativa:

- Documentales(p.e. MongoDB): guardan los datos en un JSON
- Columnares(p.e. Cassandra): los datos no se guardan en filas, sino en columnas ,ideal para tratar grandes volúmenes de datos por columnas
- Clave-Valor(p.e. DynamoDB): acceso rápido a los datos
- Graficas(p.e. Amazon Neptune): para tratar relaciones complejas



	SQL	NoSQL
<b>Estructura de datos</b>	Datos estructurados con esquema rígido (tablas, filas, columnas)	Datos semi-estructurados o no estructurados (JSON, columnas, clave-valor)
<b>Modelo de datos</b>	Relacional	Varios modelos: documental, columnares, clave-valor, gráficas
<b>Integridad de datos</b>	Alta	Variable
<b>Escalabilidad</b>	Vertical (mejorar hardware de un solo servidor)	Horizontal (añadir más servidores para distribuir carga)
<b>Consultas complejas</b>	Soportadas y optimizadas	Menos eficientes o más complicadas para consultas complejas
<b>Casos de uso típicos</b>	Sistemas con datos estructurados, transacciones, bancos, ERP	Big data, datos no estructurados, aplicaciones web, IoT, análisis en tiempo real
<b>Flexibilidad en esquema</b>	Baja	Alta
<b>Rendimiento en volumen alto</b>	Puede verse limitado por la escalabilidad vertical	Muy buena para volúmenes grandes y distribuidos

Figura 2.6: Tabla comparativa tipos de bases de datos

### 2.5.2. Backend

En el desarrollo del backend existen varios frameworks que nos van a permitir un desarrollo rápido, eficaz y de calidad. Nuestro backend se va encargar principalmente de recibir y realizar peticiones de los clientes y/o consultas a la BD. Para ello se han investigado las siguientes herramientas:

- Express.js(p.e. Node.js): muy escalable y alto rendimiento
- Django(p.e. Python): ideal para sistemas a gran escala

- Ruby on rails: el mejor en velocidad de desarrollo

Existen más frameworks, pero hemos decidido investigar solo sobre aquellos que me permitan un desarrollo más ágil, el resto de herramientas como Spring Boot(Java) o Laravel(PHP), también son buenas herramientas, pero tenemos más familiaridad con algunas de las herramientas anteriores, a continuación se muestra una tabla comparativa entre ellas.

	<b>Express.js (Node.js)</b>	<b>Django (Python)</b>	<b>Ruby on Rails</b>
<b>Lenguaje base</b>	JavaScript	Python	Ruby
<b>Rendimiento</b>	Alto, muy escalable	Muy bueno, adecuado para gran escala	Bueno, optimizado para desarrollo rápido
<b>Velocidad de desarrollo</b>	Rápida, modular y flexible	Rápida, con muchas funcionalidades integradas	Muy rápida, con convenciones que agilizan el desarrollo
<b>Facilidad de aprendizaje</b>	Fácil para desarrolladores JS	Moderada, requiere conocer Python	Moderada, requiere conocer Ruby
<b>Comunidad y soporte</b>	Amplia comunidad, mucha documentación	Comunidad sólida y madura	Comunidad activa y enfocada en productividad
<b>Casos de uso típicos</b>	APIs REST, aplicaciones en tiempo real, microservicios	Aplicaciones web completas, proyectos grandes	Proyectos con desarrollo ágil, startups, MVPs
<b>Flexibilidad</b>	Muy alta, minimalista	Completo y con muchas herramientas incluidas	Alta, con convenciones fuertes para facilitar buenas prácticas
<b>Herramientas integradas</b>	Básico, depende de módulos externos	Incluye ORM, sistema de autenticación, administración	Incluye ORM, sistema de rutas, validaciones integradas

Figura 2.7: Tabla comparativa frameworks para el backend

### 2.5.3. Frontend

Como se dijo en la introducción, uno de los objetivos es que el software sea multiplataforma, así que se investigará sobre frameworks que me permitan esa capacidad, después se muestra una tabla comparativa:

- Flutter(lenguaje Dart): de las manos de Google, esta herramienta permite un desarrollo rápido ya que nos permite el hot reload y no tener que recom-

pilar la app cada vez que haya un cambio. También es muy personalizable en lo que respecta a la UI.

- React Native(lenguaje JavaScript): es un framework que extiende React, permitiendo hacer sentir aplicaciones en js como si fueran nativas.
- Xamarin(Lenguajes C#): de Microsoft, esta herramienta es ideal para desarrolladores familiarizados con C# y .NET, también es idóneo para el alto rendimiento y acceso a HW nativo.

	Flutter (Dart)	React Native (JavaScript)	Xamarin (C#)
Lenguaje base	Dart	JavaScript (extensión de React)	C# (.NET)
Velocidad de desarrollo	Alta, con hot reload para ver cambios al instante	Alta, con hot reload	Moderada, requiere compilación
Experiencia UI	Muy personalizable, widgets propios para UI nativa	UI basada en componentes nativos	Acceso a controles nativos para UI
Rendimiento	Muy cercano a nativo	Bueno, aunque puede depender de puentes JS nativos	Muy cercano a nativo, alto rendimiento
Facilidad de aprendizaje	Requiere conocer Dart	Fácil si se conoce JavaScript y React	Más fácil para desarrolladores C#.NET
Acceso a hardware nativo	Sí, mediante plugins	Sí, mediante módulos nativos	Sí, acceso directo al hardware nativo
Comunidad y soporte	Creciente, fuerte respaldo de Google	Amplia comunidad y ecosistema React	Comunidad sólida dentro del ecosistema Microsoft
Casos de uso típicos	Aplicaciones móviles con UI personalizada y multiplataforma	Apps móviles multiplataforma con rapidez y flexibilidad	Apps multiplataforma con integración profunda en Windows y otras plataformas

Figura 2.8: Tabla comparativa frameworks para el frontend

# Capítulo 3

## Propuesta

### 3.1. Descripción detallada

La app a desarrollar daría la mayoría de funcionalidades ofertadas por las aplicaciones de pago de una manera gratuita y aporta su funcionalidad de medir de forma personalizada el rendimiento del deportista. También se aporta la funcionalidad de planificar los entrenamientos de tu próximo mes y ver el rendimiento de los entrenamientos previamente realizados, todo esto interactuando con un calendario en el menú principal. Otra funcionalidad que se aporta es la de compartir rutinas entre usuarios. Si fuese necesario ver más a fondo el rendimiento en un ejercicio concreto se puede hacer una gráfica con las marcas obtenidas en los entrenamientos. Las rutinas se trataran como grupos de ejercicios que pueden ser asignadas a días concretos en el calendario. Durante el entrenamiento para llevar el control del ejercicio se pedirá al usuario sobre las marcas obtenidas y de paso no se permitirá seguir hasta completar el descanso estipulado en la rutina por el usuario, esto para que se entrene de manera adecuada y prevenir lesiones. El sistema para detectar cumplir metas se usa justo al terminar el entrenamiento, revisa todos los datos obtenidos de mientras se van guardando en memoria y le lanza un display al usuario avisando de las metas cumplidas en cada ejercicio. En la app la inmensa mayoría de los datos se guardan de manera local y los únicos datos que pueden salir del dispositivo son las rutinas que se suben en la nube bajo petición expresa del usuario. La memoria local estará separada entre usuarios permitiendo la existencia de varios en un mismo dispositivo.

## 3.2. Elección de tecnologías

## 3.3. Backend

Se usará Express.js(Node.js), la principal razón que se tiene para optar por esta tecnología, es la gran familiaridad que se tiene con el entorno de JavaScript, se conocen las tecnologías y la forma de implementación, su gran flexibilidad para implementar muchas funcionalidades diversas también fué una buena baza para escoger esta herramienta. Django fué la primera descartada, porque como bien dice en la tabla de la sección anterior(fig. 2.7) está más enfocada a proyectos de gran envergadura. Ruby on Rails fue la que más dudas planteó, como se dijo anteriormente es muy buena opción para las metodologías ágiles y como se dirá más adelante, se optará por una metodología ágil para el desarrollo. Pero al final se descartó por el desconocimiento de la herramienta y la falta de tiempo para aprender bien una nueva tecnología de forma que se pueda usar con efectividad.

## 3.4. Frontend

Se ha decidido usar Flutter, dado que Xamarin es buena herramienta, pero está más enfocada a un desarrollo en un ambiente en el cuál solo se usan herramientas de Microsoft, siendo complicada la conectividad y soporte entre herramientas de distintas empresas, esto nos cierra el abanico de posibilidades. React Native también es una buena herramienta, fue principalmente la que más dudas sembró, ya que como se dijo previamente se usará un backend basado en Express.js, esto entonces permitiría unificar entornos lo cual favorecería al desarrollo. pero se descartó por lo personalizable que son las IUs desarrolladas en Flutter respecto a esta tecnología.

## 3.5. Base de datos

Para las bases de datos, tanto para el backend como para la local de la app se usará una SQL, dado a la familiaridad que se dispone con esta tecnología y la necesidad de unificar el tipo de tecnología del servidor y de la app. Esto anterior se debe a que como bien se ha dicho se usará Flutter como entorno frontend y la herramienta que se facilita para el almacenamiento local en este es SQLite, basada en un modelo relacional.

Los modelos NoSQL, se descartaron principalmente porque las relaciones entre elementos suelen ser más complejas y en este proyecto es necesario que los datos se traten con una estructura determinada y simple.

### 3.6. Diagrama de arquitectura

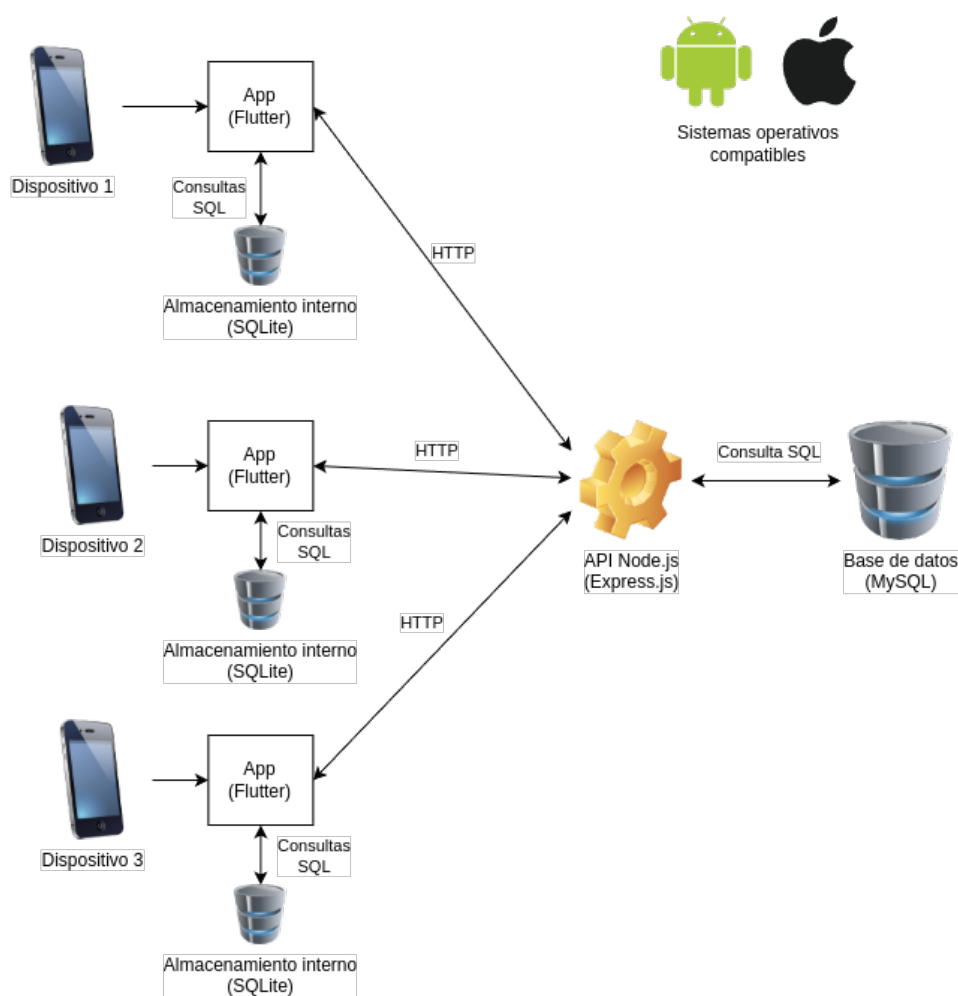


Figura 3.1: Diagrama de arquitectura

### 3.7. Metodología utilizada

Para el desarrollo de esta app, se usará una metodología *ágil* tipo **SCRUM**. Se decidió usar esta porque permite corregir fallos en la velocidad de diseño y/o planificación de forma eficiente y sin dañar el producto final. Esta metodología permite también entregar desarrollos funcionales a la tutora e ir validando las funcionalidades poco a poco mediante reuniones.

Las reuniones con la tutora serán las *sprint reviews*.

### 3.8. Temporización

La temporización se realizó el día 25 de marzo de 2025. La entrega del producto (este TFG) está prevista para el 16 de junio de 2025, es decir, 83 días, o lo que es lo mismo, casi 12 semanas. Si un sprint dura 2 semanas, habrá 6 sprints hasta la entrega final.

La iteración 0 se dedicará al diseño de pantallas y al repaso de las funcionalidades de la app, para concretar las historias de usuario y sus prioridades. La idea inicial es dividir la app en varios módulos y centrar cada sprint en cada uno de ellos:

1. Ejercicios
2. Rutinas, usuarios y sesión
3. Descargar y compartir rutinas
4. Flujo de entrenamiento
5. Inteligencia Artificial (IA)
6. Smartwatch y tratamiento de datos

Al final no se pudo cumplir con la temporización porque se dieron una serie de problemas durante el desarrollo, tanto con algunas tecnologías que en la versión final no se implementaron como intentando aprender sobre ellas.

### 3.9. Desarrollo

Para cada iteración se inicia una fase de diseño y aprendizaje sobre las herramientas que se vayan a usar. Al final de cada iteración se añade la contenido a la documentación especificando lo desarrollado. Después del sprint review se añade su consecuente parte especificando lo corregido y discutido en dicha reunión. Aquí se aporta un breve resumen:

- Iteración 0: diseños de las principales pantallas para explicar que se pretende desarrollar a la tutora y especificar las HU



- Iteración 1: desarrollo de la lista de ejercicios
- Iteración 2: desarrollo de la lista de rutinas y sistema de sesiones de usuario
- Iteración 3: desarrollo de la funcionalidad de compartir rutinas en la nube
- Iteración 4: desarrollo del calendario y medición del entrenamiento del usuario
- Iteración 5: desarrollo de la creación de gráficas y display del rendimiento del usuario.

### 3.9.1. Iteración 0

En esta primera iteración nos centramos en realizar los diseños de la app, considerando su accesibilidad. También concreté el *product backlog*, compuesto por 41 historias de usuario. En las historias no menciono ningún actor puesto que solo existe el usuario que realiza los ejercicios y el es protagonista en todas ellas, no hay ningún moderador.

A pesar de trabajar con SCRUM se realizarán todos los diseños de interfaces de usuario en este primer sprint para concretar con la tutora los requisitos principales de la aplicación a desarrollar en este TFG. En nuestro caso los diseños de las interfaces se usarán como herramientas de captura y especificación de requisitos. En cada iteración se podrán revisar si hay cambios.

La lista inicial de historias de usuario es la siguiente, y algunas incluyen tareas secundarias:

- SCRUM-1** Registrar peso por día
- SCRUM-2** Establecer peso objetivo
- SCRUM-3** Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
- SCRUM-4** Buscar rutina en la lista del usuario
- SCRUM-5** Insertar/Borrar/Modificar rutina
- SCRUM-6** Hacer gráfica en base a las marcas obtenidas
- SCRUM-7** Revisar datos para ver si el descanso es necesario
- SCRUM-8** Enseñar datos de una rutina a descargar
- SCRUM-9** Compartir mi rutina
- SCRUM-10** Valorar el entrenamiento en base a la marca actual y la meta del usuario
- SCRUM-11** Monitorizar pulso en tiempo real
- SCRUM-12** Medir pulso en reposo y compararlo con datos de ejercicios
- SCRUM-13** Avisar de anomalías en el pulso de forma suave
- SCRUM-14** Obtener calorías quemadas
- SCRUM-15** Comprobar el equilibrio nervioso del usuario
- SCRUM-16** Realizar el flujo del entrenamiento
- SCRUM-17** Conectar con la IA para iniciar diálogo
- SCRUM-18** Crear/Borrar usuario
- SCRUM-19** Resumir datos
- SCRUM-20** Iniciar/Cerrar sesión

**SCRUM-21** Medir SpO2

**SCRUM-22** Interpretar constantes

**SCRUM-23** Buscar ejercicios en la lista de ejercicios

**SCRUM-24** Pop up de confirmación

**SCRUM-25** Implementar menú principal

**SCRUM-26** Detectar metas cumplidas en los ejercicios después del entrenamiento

**SCRUM-27** Buscar rutinas para descargar

**SCRUM-28** Implementar calendario

A continuación se muestran los diseños creados en esta iteración con la herramienta web figma:

## Inicio de sesion y crear usuario

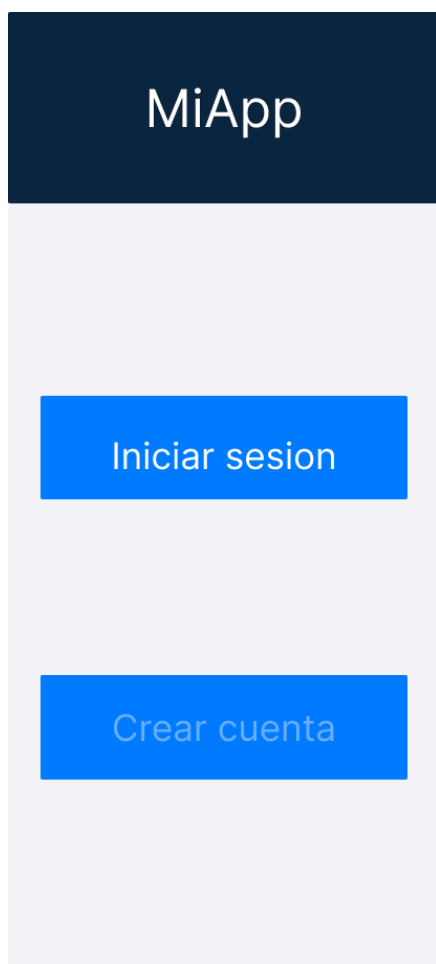


Figura 3.2: Pagina inicial

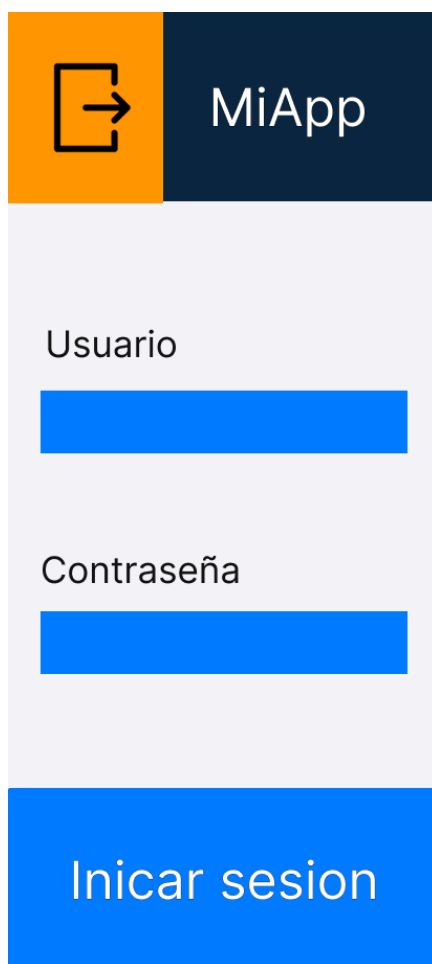



Figura 3.3: Inicio de sesion

 MiApp

Correo


Usuario

Contraseña

Repite contraseña

Seguir

Figura 3.4: Crear cuenta 1

 MiApp

Fecha Nacimiento:

Género:

Peso:

Altura:

Terminar

Figura 3.5: Crear cuenta 2

### Menu principal

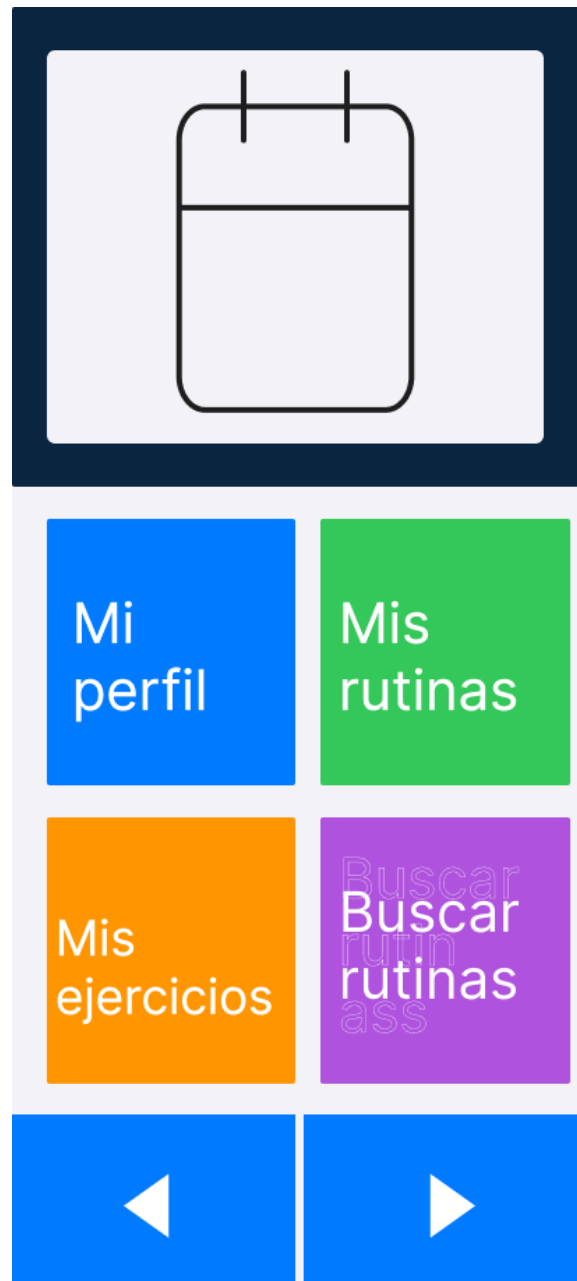


Figura 3.6: Menu principal

## Lista ejercicios del usuario

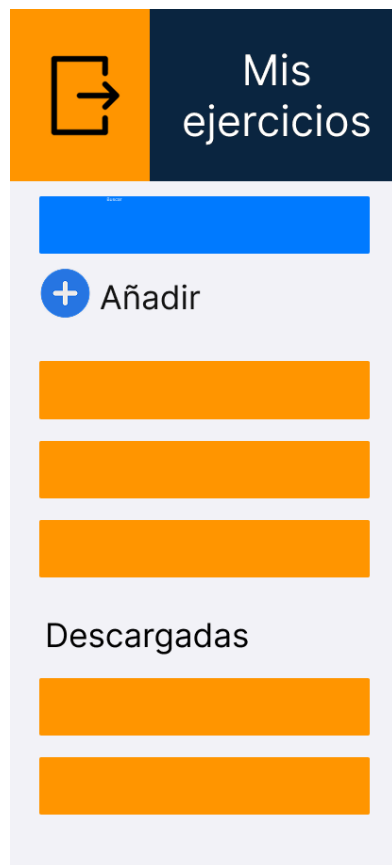


Figura 3.7: Lista ejercicios

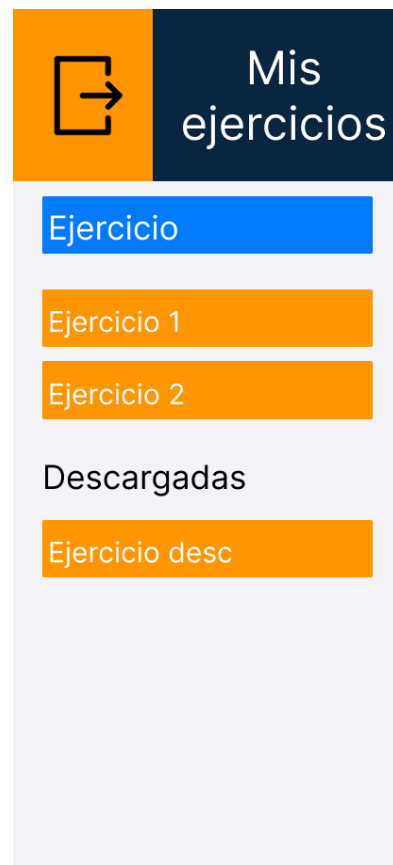


Figura 3.8: Lista ejercicios filtrada

Nombre ejercicio

Marca actual:  
7.4 repeticiones x 6kg

Valoración actual: 4.5 / 5

Graficar según tiempo

Meta actual:  
8 repeticiones x 6kg

Sin cumplir

Establecer meta

Figura 3.9: Datos ejercicio



Figura 3.10: Pop up graficar según tiempo

Repeticiones

Peso

Guardar

Figura 3.11: Pop up establecer meta

Añadir ejercicio

Nombre

¿Que se mide?

Guardar

Figura 3.12: Pop up crear ejercicio

### Lista rutinas

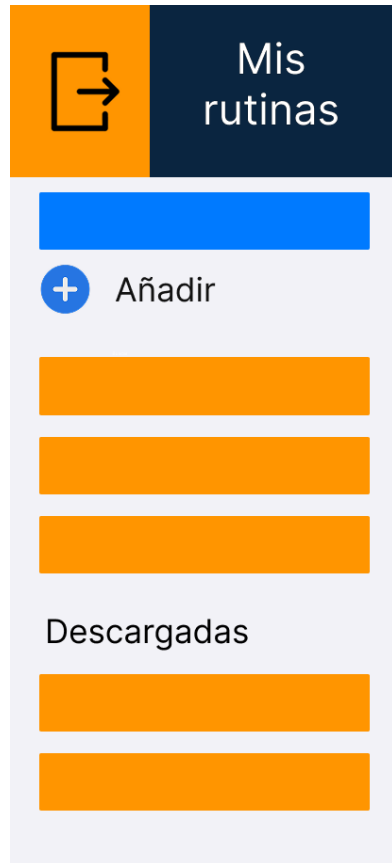


Figura 3.13: Lista rutinas

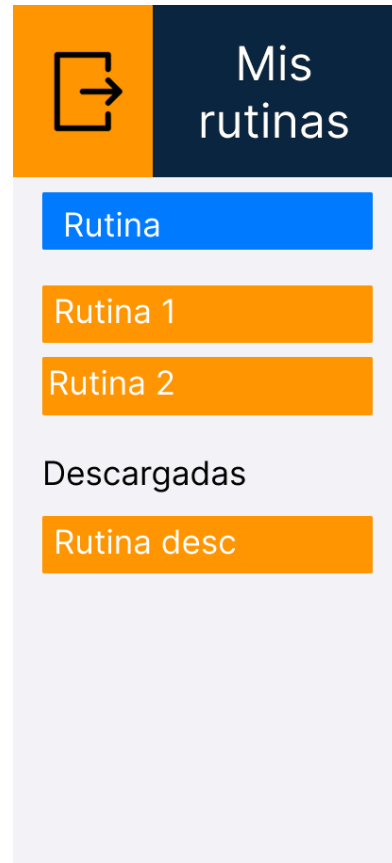


Figura 3.14: Lista rutinas filtradas



Figura 3.15: Datos rutina modificable



Figura 3.16: Datos rutina no modificable





Figura 3.17: Modificar ejercicios rutina

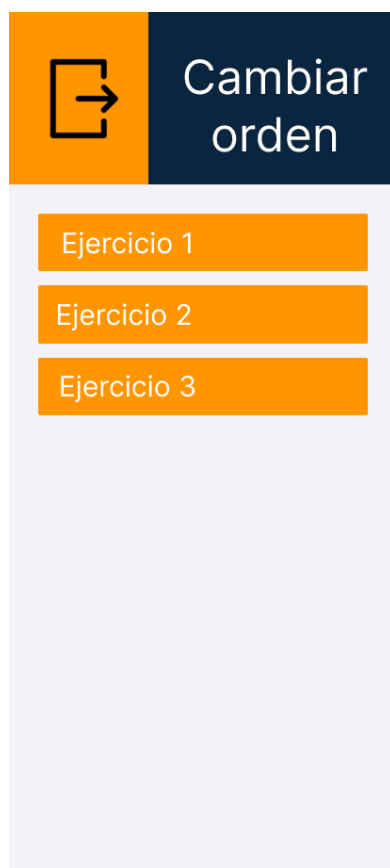


Figura 3.18: Cambiar orden ejercicios rutina

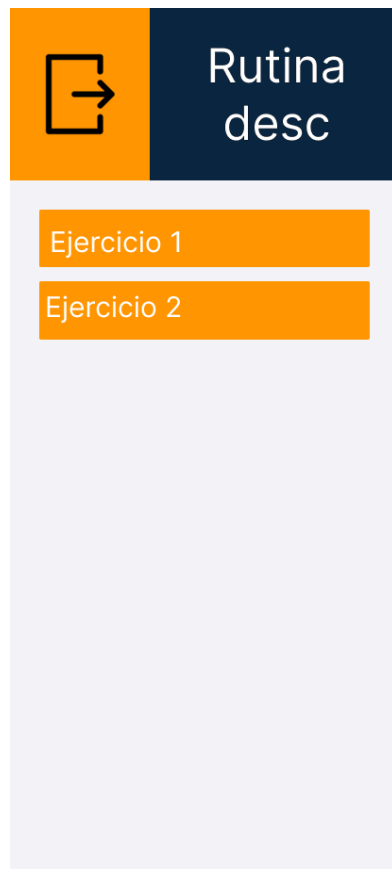


Figura 3.19: Ejercicios rutina descargada

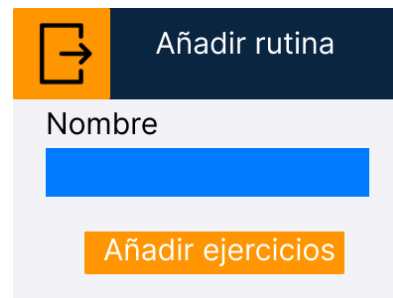


Figura 3.20: Pop up crear rutinas

## Mi perfil

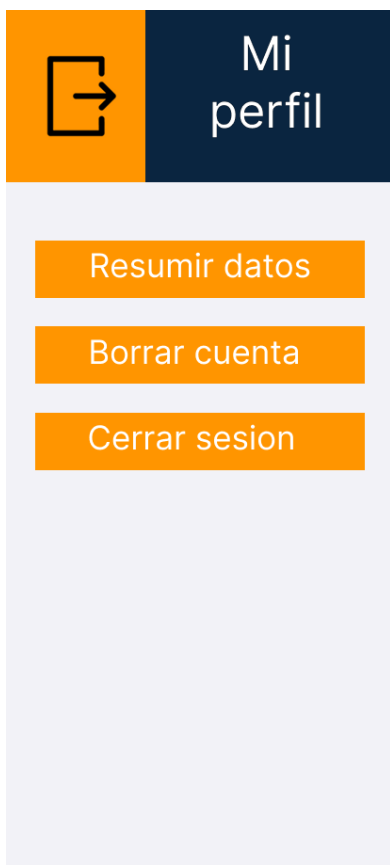


Figura 3.22: Pop up de confirmacion

Figura 3.21: Opciones de perfil usuario

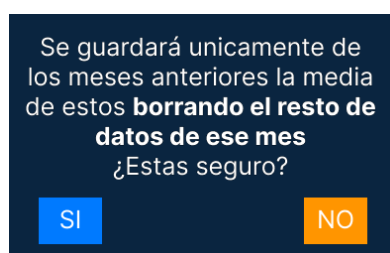


Figura 3.23: Pop up de confirmacion  
resumir datos

Buscar rutina para descargar

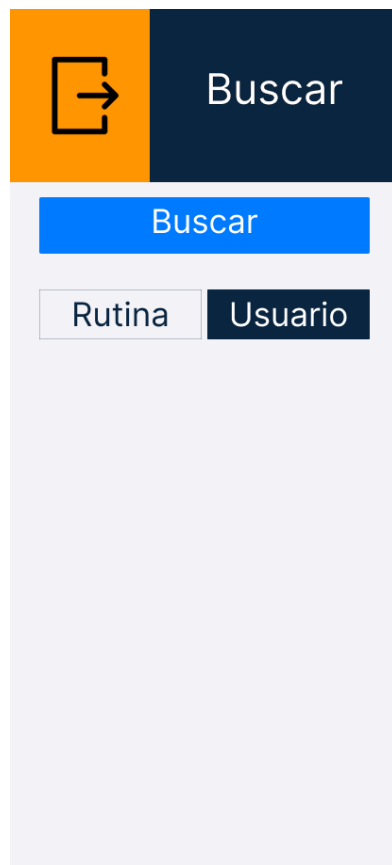


Figura 3.24: Buscar usuario



Figura 3.25: Buscar usuario filtrado



Figura 3.26: Rutinas de un usuario

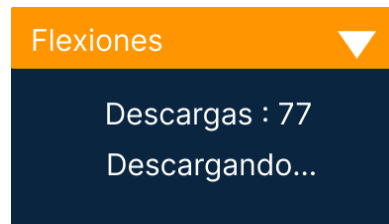


Figura 3.27: Widget descargar rutina

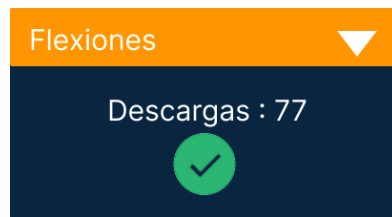


Figura 3.28: Widget rutina descargada



Figura 3.29: Buscar rutinas por su nombre filtrada

Entrenamiento (Una vez seleccionada una fecha en el calendario del menu principal saldría la siguiente ventana)

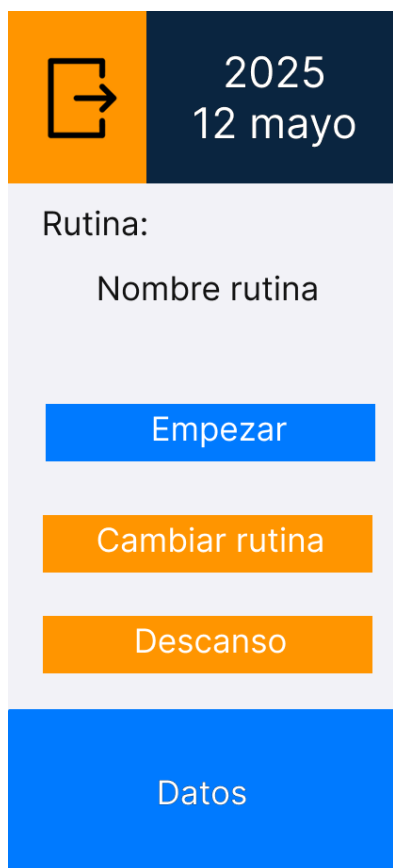


Figura 3.30: Entrenamiento de un día determinado

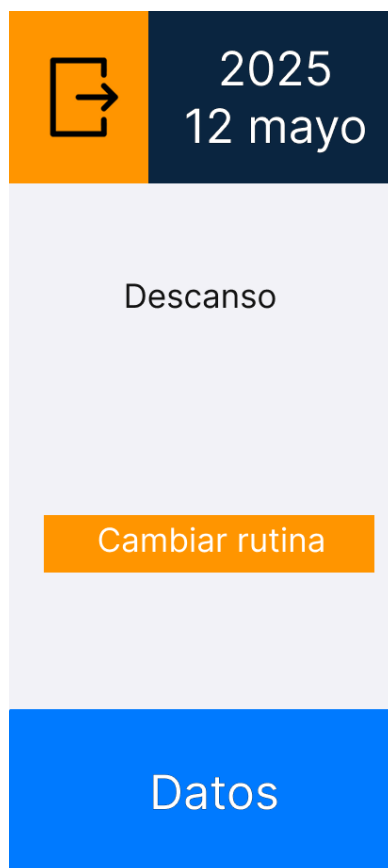


Figura 3.31: Descanso en un día determinado

Si seleccionamos datos de una fecha saldría esto

The image shows a mobile application interface. At the top, there is a header bar with two sections: an orange section on the left containing a white icon of a square with an arrow pointing right, and a dark blue section on the right containing the text "2025" and "12 mayo" in white. Below the header, the main content area has a light gray background. It contains two sections. The first section has the text "Peso corp: sin ingresar" in black, followed by an orange button with the text "Ingresar" in white. The second section has the text "Composicion corporal:" in black, followed by "Músculo: X% Grasa: X% Osea: X%" in black, and then another orange button with the text "Ingresar" in white.

Figura 3.32: Frame 29.png



La siguiente pantalla sale al comenzar el entrenamiento

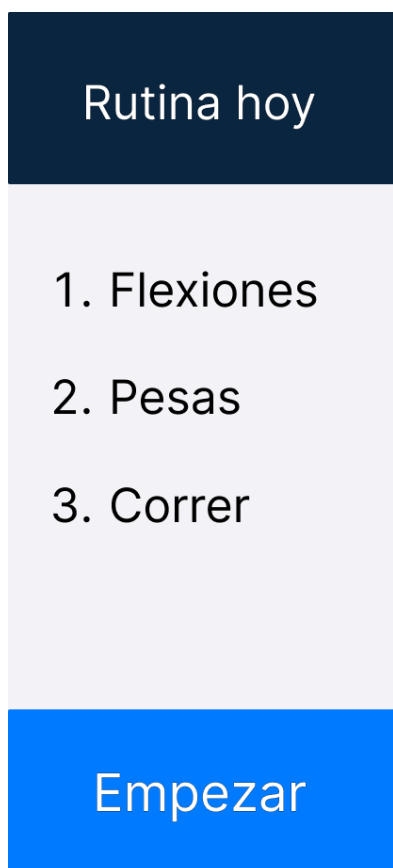


Figura 3.33: Lista ejercicios del entrenamiento actual

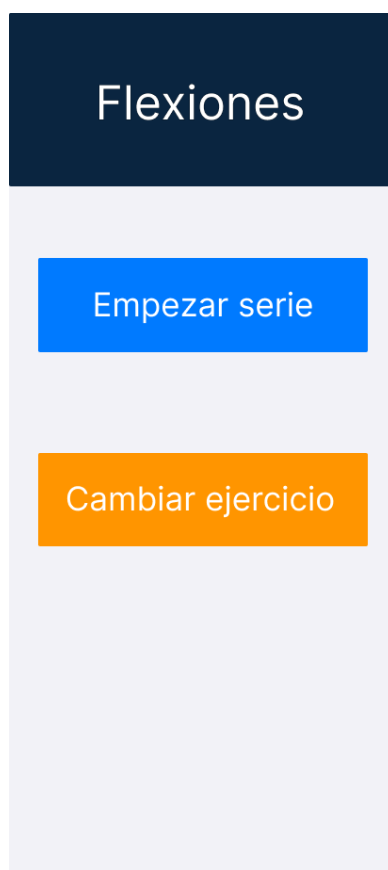


Figura 3.34: Primera serie flexiones

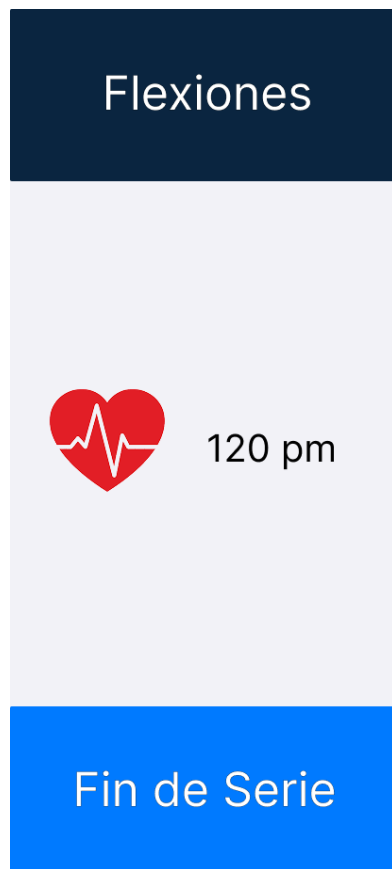


Figura 3.35: Realizando flexiones

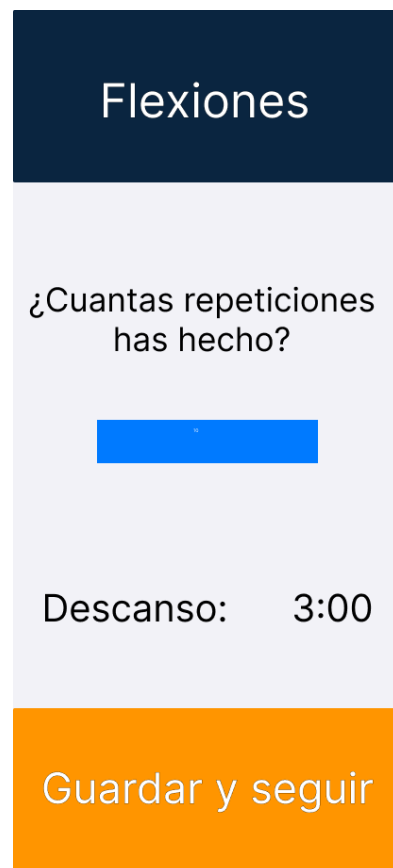


Figura 3.36: Fin de la serie (Descanso no completado)

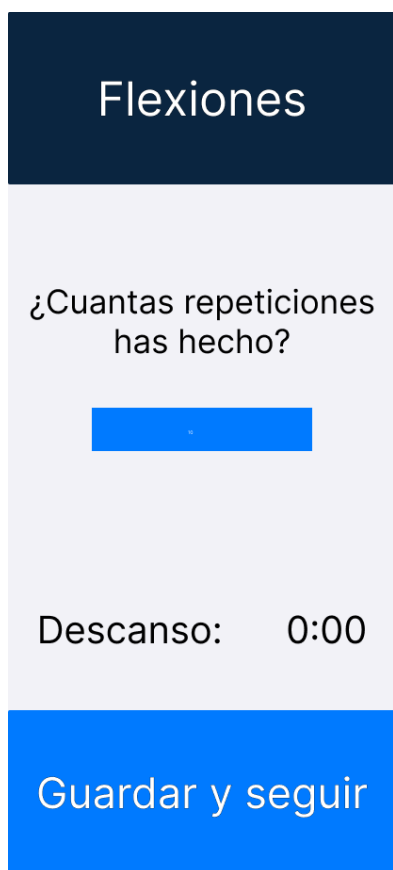


Figura 3.37: Fin de la serie (Descanso completado)

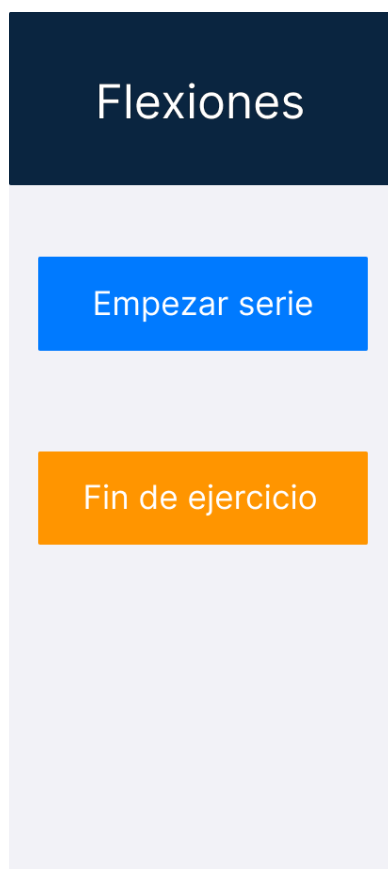


Figura 3.38: Acabar ejercicio o añadir serie

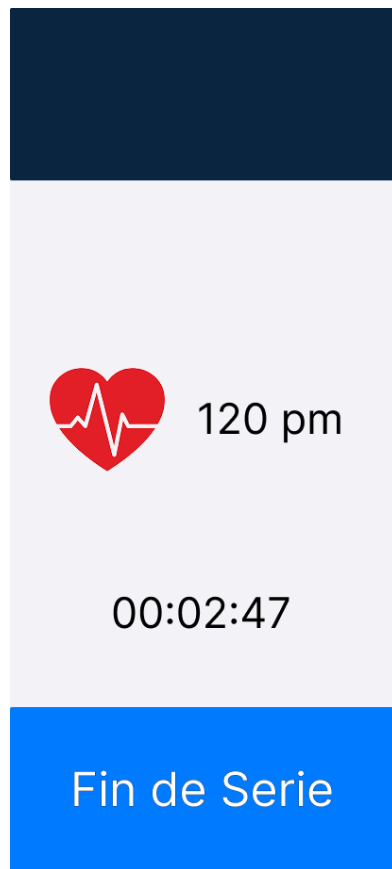


Figura 3.39: Realizando ejercicio midiendo tiempo

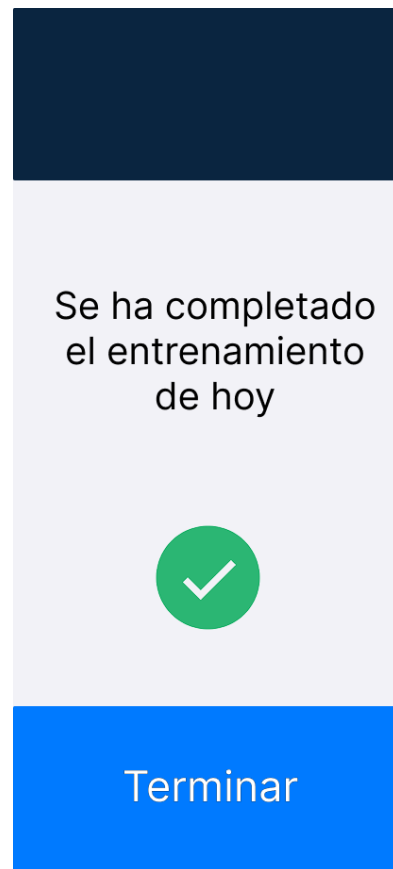


Figura 3.40: Fin entrenamiento



Figura 3.41: Metas cumplidas

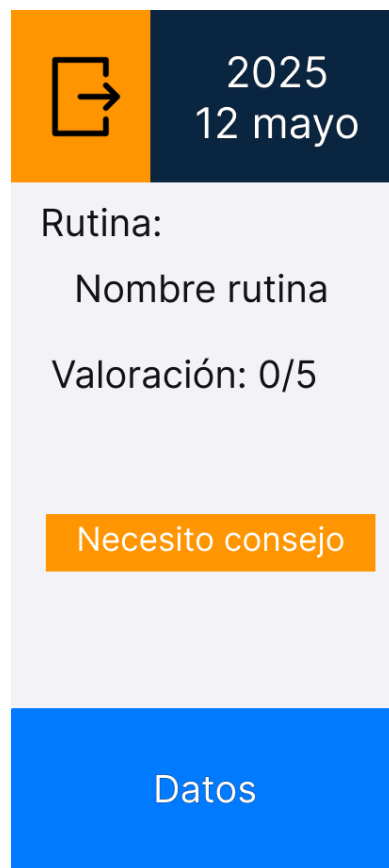


Figura 3.42: Datos entrenamiento terminado

El chat de la IA se abre cuando se le da a la opción necesito consejo de un entrenamiento realizado

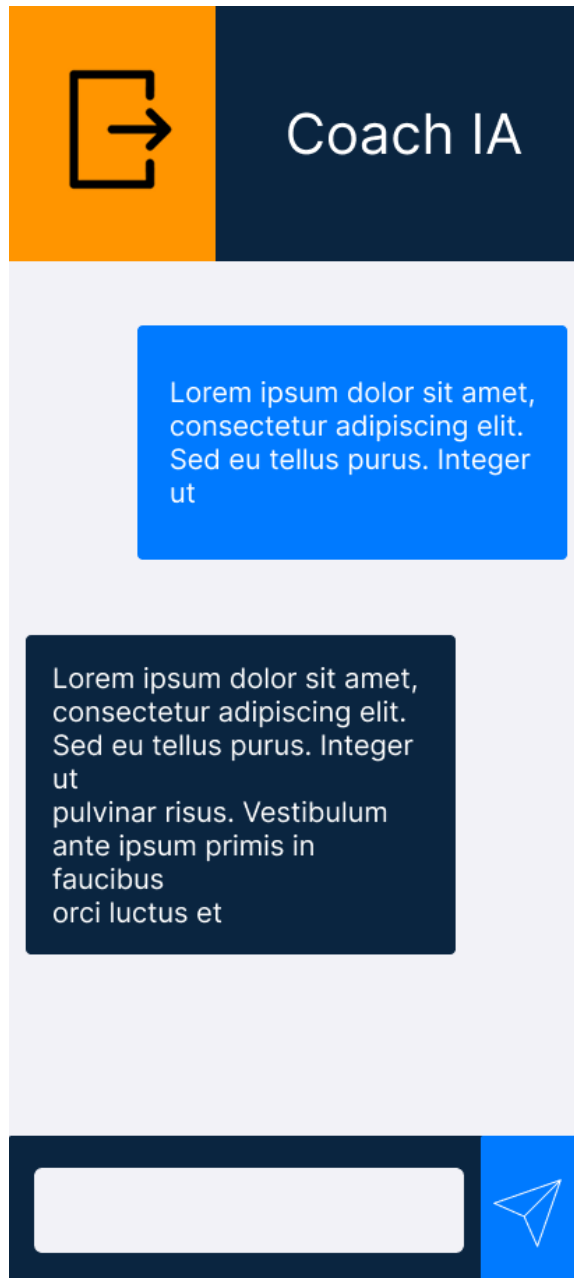


Figura 3.43: Chat con IA

Si se selecciona a datos de un entrenamiento terminado

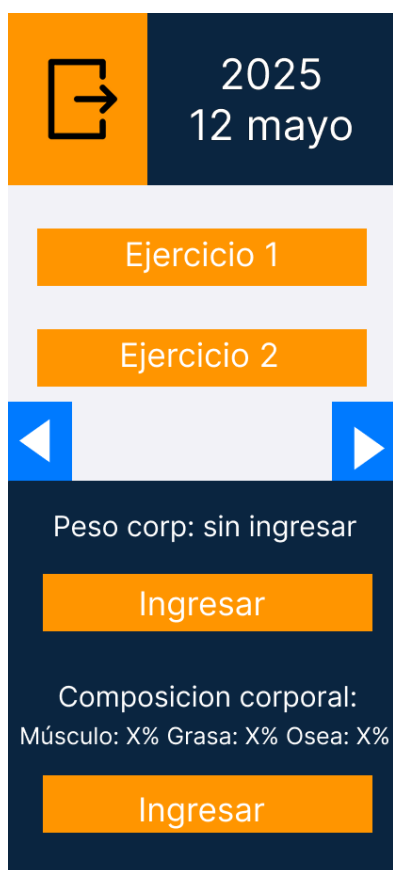


Figura 3.44: Datos detallados entrenamiento terminado

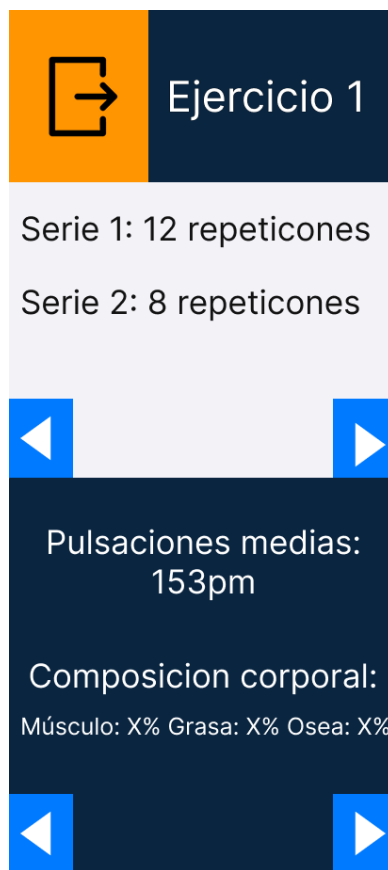


Figura 3.45: Pop up datos de un ejercicio en entrenamiento

Como puede observarse, para cubrir las guías de accesibilidad, el tamaño de la letra es grande (pon tamaño medio en puntos) para facilitar la lectura por personas con problemas de visión. El tamaño de los botones y elementos seleccionables también es grande, mayor de 9 milímetros que es lo estipulado, con suficiente espacio entre ellos (más de 2 milímetros). El contraste de color entre el texto, imágenes y fondo es aceptable, tal y como también recomiendan las guías.<sup>i</sup>

Estos diseños pueden cambiar y sufrir alteraciones debido a los sprints reviews y correcciones de accesibilidad. De todas formas, cada cambio será informado con su motivo.

#### 3.9.1.1. Sprint Review 0

Durante esta revisión de sprint se realizaron mejoras de diseño, como la incorporación de iconos en todos los botones para mejorar la accesibilidad. Se revisaron también los títulos de las ventanas, cambiando aquellos que no eran lo suficientemente claros.

Además, se añadieron nuevas historias al *product backlog*, incorporando o desglosando funcionalidades:

- SCRUM-29: Copiar rutina
- SCRUM-30: Añadir meta por parámetro
- SCRUM-31: Solicitar permiso al usuario antes de enviar datos a la IA
- SCRUM-32: Enviar datos del entrenamiento actual y anteriores a la IA
- SCRUM-33: Conectar/Desconectar con la IA

El backlog quedaría de la siguiente manera:



ID	Descripción	Prioridad (MoSCoW)
SCRUM-1	Registrar peso por día	C
SCRUM-2	Establecer peso objetivo	W
SCRUM-3	Insertar/Borrar/Modificar ejercicio de la lista de ejercicios	M
SCRUM-4	Buscar rutina en la lista del usuario	M
SCRUM-5	Insertar/Borrar/Modificar rutina	M
SCRUM-6	Hacer gráfica en base a las marcas obtenidas	C
SCRUM-7	Revisar datos para ver si el descanso es necesario	W
SCRUM-8	Enseñar datos de una rutina a descargar	S
SCRUM-9	Compartir mi rutina	M
SCRUM-10	Valorar el entrenamiento en base a la marca actual y la meta del usuario	W
SCRUM-11	Monitorizar pulso en tiempo real	C
SCRUM-12	Medir pulso en reposo y compararlo con datos de ejercicios	W
SCRUM-13	Avisar de anomalías en el pulso de forma suave	C
SCRUM-14	Obtener calorías quemadas	W
SCRUM-15	Comprobar el equilibrio nervioso del usuario	W
SCRUM-16	Realizar el flujo del entrenamiento	M
SCRUM-17	Conectar con la IA para iniciar diálogo	C

Figura 3.46: Backlog prioridades MoSCoW

SCRUM-18	Crear/Borrar usuario	M
SCRUM-19	Resumir datos	W
SCRUM-20	Iniciar/Cerrar sesión	M
SCRUM-21	Medir SpO2	W
SCRUM-22	Interpretar constantes	C
SCRUM-23	Buscar ejercicios en la lista de ejercicios	M
SCRUM-24	Pop up de confirmación	M
SCRUM-25	Implementar menú principal	M
SCRUM-26	Detectar metas cumplidas en los ejercicios despues del entrenamiento	S
SCRUM-27	Buscar rutinas para descargar	M
SCRUM-28	Implementar calendario	M
SCRUM-29	Copiar rutina	C
SCRUM-30	Añadir meta por parámetro	C
SCRUM-31	Enseñar las marcas obtenidas en un día	M
SCRUM-32	Solicitar permiso al usuario antes de enviar datos a la IA	W
SCRUM-33	Enviar datos del entrenamiento actual y anteriores a la IA	S

Figura 3.47: Backlog prioridades MoSCoW

ID	Descripción	Prioridad (MoSCoW)
SCRUM-23	Buscar ejercicios en la lista de ejercicios	M
SCRUM-3	Insertar/Borrar/Modificar ejercicio de la lista de ejercicios	M
SCRUM-24	Pop up de confirmación	M
SCRUM-30	Añadir meta por parámetro	C

Figura 3.48: Iteracion 1

ID	Descripción	Prioridad (MoSCoW)
SCRUM-18	Crear/Borrar usuario	M
SCRUM-20	Iniciar/Cerrar sesión	M
SCRUM-5	Insertar/Borrar/Modificar rutina	M
SCRUM-4	Buscar rutina en la lista del usuario	M
SCRUM-25	Implementar menú principal	M

Figura 3.49: Iteracion 2

ID	Descripción	Prioridad (MoSCoW)
SCRUM-2	Establecer peso objetivo	W
SCRUM-9	Compartir mi rutina	M
SCRUM-27	Buscar rutinas para descargar	M
SCRUM-8	Enseñar datos de una rutina a descargar	S
SCRUM-7	Revisar datos para ver si el descanso es necesario	W
SCRUM-19	Resumir datos	W
SCRUM-29	Copiar rutina	C

Figura 3.50: Iteracion 3

ID	Descripción	Prioridad (MoSCoW)
SCRUM-26	Detectar metas cumplidas en los ejercicios despues del entrenamiento	S
SCRUM-1	Registrar peso por día	C
SCRUM-16	Realizar el flujo del entrenamiento	M
SCRUM-28	Implementar calendario	M
SCRUM-10	Valorar el entrenamiento en base a la marca actual y la meta del usuari	W
SCRUM-31	Enseñar las marcas obtenidas en un día	M
SCRUM-6	Hacer gráfica en base a las marcas obtenidas	S

Figura 3.51: Iteracion 4

ID	Descripción	Prioridad (MoSCoW)
SCRUM-32	Solicitar permiso al usuario antes de enviar datos a la IA	W
SCRUM-33	Enviar datos del entrenamiento actual y anteriores a la IA	S
SCRUM-17	Conectar con la IA para iniciar diálogo	W

Figura 3.52: Iteracion 5

ID	Descripción	Prioridad (MoSCoW)
SCRUM-11	Monitorizar pulso en tiempo real	C
SCRUM-12	Medir pulso en reposo y compararlo con datos de ejercicios	W
SCRUM-13	Avisar de anomalías en el pulso de forma suave	C
SCRUM-14	Obtener calorías quemadas	W
SCRUM-15	Comprobar el equilibrio nervioso del usuario	W
SCRUM-21	Medir SpO2	W
SCRUM-22	Interpretar constantes	C

Figura 3.53: Iteracion 6

### 3.9.2. Iteración 1

La lista de historias de la primera iteracion queda de la siguiente manera:

- SCRUM-23: Buscar ejercicio en lista de ejercicios
- SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
- SCRUM-24: Pop up de confirmación
- SCRUM-30: Añadir meta por parámetro

A continuación especificaremos las tareas y pruebas de cada historia, y los resultados asociados a cada una.

SCRUM-23: Buscar ejercicio en lista de ejercicios
Tareas: <ol style="list-style-type: none"><li>1. Implementar la BD local</li><li>2. Implementar algoritmo de búsqueda por nombre</li><li>3. Implementar boceto de IU fig. 3.7</li></ol>
Pruebas de aceptación: <ul style="list-style-type: none"><li>■ si no existe ejercicio buscado, no sale nada en pantalla</li><li>■ si existe ejercicio buscado, se muestra el ejercicio buscado</li></ul>

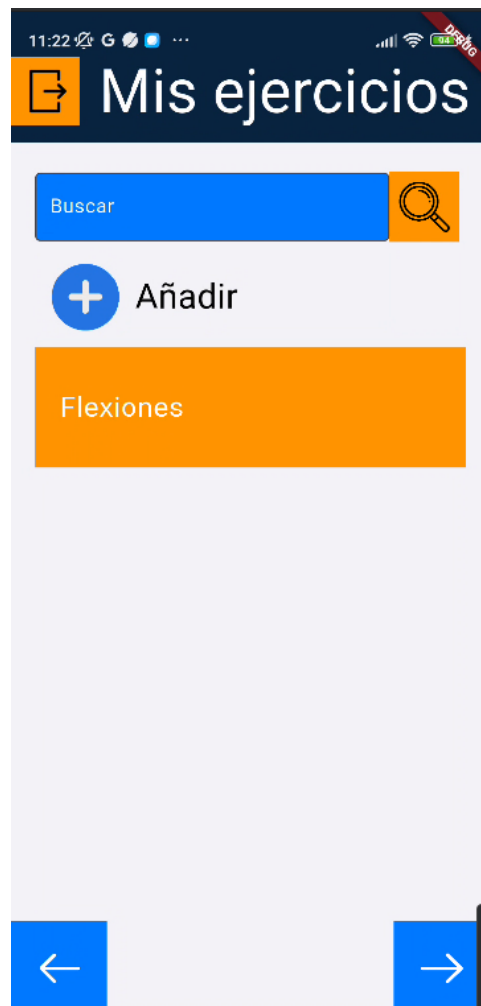


Figura 3.54: Resultado de la implementación de la lista de ejercicios

El algoritmo de búsqueda funciona de la siguiente manera, digamos que se busca el ejercicio 'flexiones' en la lista, entonces se pasan a minúscula todas las cadenas de caracteres de la lista, se pasa la cadena 'flexiones' a minúscula

y se obtiene como resultado los elementos de la lista que contengan la cadena 'flexiones'.

Una de las dificultades de esta funcionalidad fue que esta pantalla debería funcionar como una plantilla, es decir, la lista de elementos como una lista de Strings y la forma de acceder a cada elemento, de esta forma se reciclaría mucho código ahorrando tiempo. Lo que se hizo fue una ingeniería inversa, se desarrolló la pantalla forma específica para los ejercicios y cuando funcionaba correctamente, se fue abstrayendo el código hasta obtener la clase ListaBusquedaAniadir, la cual tiene el comportamiento deseado de una plantilla.

Otro problema encontrado durante las pruebas fue la actualización de la lista, dado que en flutter si se tiene una instancia de un widget llamémoslo A, que forma parte de otro llamado B, si se llama a una actualización de la IU desde B porque ha cambiado el contenido se genera una excepción. Como solución, se optó por solo usar la función setState (Usada para actualizar el contenido) dentro del código de la plantilla y cada vez que fuera necesario hacer una actualización desde B recargar otra vez la pantalla entera.

Esta funcionalidad pasó todas las pruebas de aceptación planificadas.

SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar la BD local</li> <li>2. Implementar las funciones para las consultas en la BD</li> <li>3. Implementar las funciones para la insercción en la BD</li> <li>4. Implementar las funciones para las modificaciones</li> <li>5. Implementar las funciones para el borrado</li> <li>6. Implementar boceto de la IU fig. 3.9, para borrar/modificar desde ahí</li> </ol>
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se borra un ejercicio se elimina de la lista</li> <li>■ si se modifica se actualiza el ejercicio con los nuevos datos</li> <li>■ si se crea un ejercicio se inserta en la lista de ejercicios</li> <li>■ se enseñan los datos del ejercicio de forma correcta una vez creado el ejercicio</li> </ul>

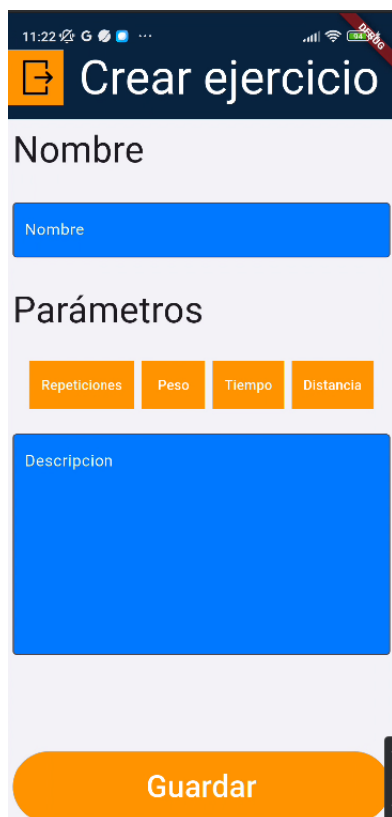


Figura 3.55: Crear ejercicios

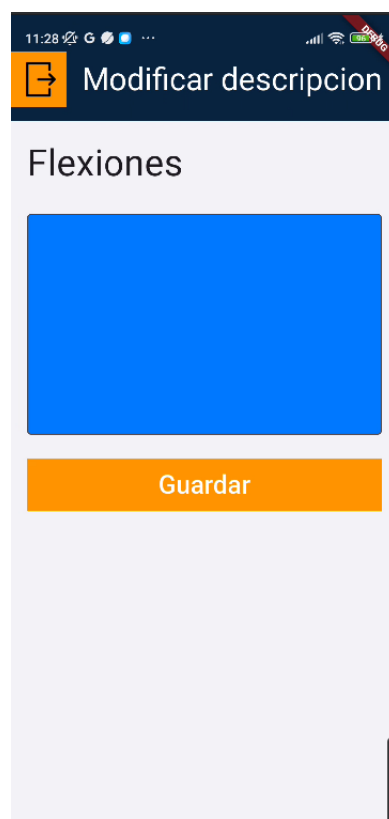


Figura 3.56: Modificar descripción ejercicios

Cuando se crea un ejercicio para seleccionar los parámetros a medir se pulsan a los botones de la sección parámetros. Los parámetros de color naranja no se miden. Mínimo un ejercicio tiene que medir un parámetro.

	Marca actual	Meta	
REPETICIONES	20.0	60.0	✗
PESO	20.8	10.5	✓
TIEMPO	05:00	00:20	✓
DISTANCIA	500.0 M	32.9 M	✓

Hacer grafica    Descripcion    Nueva meta

Figura 3.57: Pantalla donde se muestran los datos del ejercicio al acceder desde la lista

Pasó todas las pruebas de aceptación planificadas.



SCRUM-30: Añadir meta por parámetro
Tareas:
1. Implementar la BD local 2. Implementar las funciones para modificar metas
Pruebas de aceptación:
■ si existe una meta determinada para ese ejercicio y se quiere insertar una, sustituir la actual ■ si se insertan los valores en formato incorrecto, informar de errores ■ si se insertan los valores en formato correcto, guardar



Figura 3.58: Nueva meta

Podemos establecer metas para cualquier parámetro y no es obligatorio ponerlas. A esta pantalla se llega seleccionando el botón nueva meta de la pantalla

anterior. Cuando asigna una meta se hará visible en la pantalla de la HU anterior.

El problema de esta funcionalidad está en interpretar si un usuario quiere superar su meta u obtener una marca menor. Al final se optó por solo interpretar que se pueda superar la marca.

Cabe aclarar una cosa respecto a los formatos de los datos guardados en las metas y las marcas, se guardan 4 tipos de datos repeticiones (entero), peso (decimal), tiempo (tiempo) y distancia (decimal). Para ahorrar espacio en el almacenamiento local, ya que habra mucho volumen ocupado en memoria si el usuario sigue usando la app, los datos en formato decimal se suben como un entero, se hace la conversion  $\text{numeroEntero} = (\text{numeroDecimal} * 100).toInt()$  se guarda en ese formato y luego al sacarlo de memoria se le hace la conversión inversa.

Pasó todas las pruebas de aceptación planificadas.

SCRUM-24: Pop up de confirmación
Tareas:
<ol style="list-style-type: none"> <li>1. Implementar la confirmación para que pueda ser usado en pantallas distintas</li> <li>2. Implementar boceto de IU fig. 3.22</li> </ol>
Pruebas de aceptación:
<ul style="list-style-type: none"> <li>■ si se ha seleccionado una opcion, se devuelve la opcion seleccionada</li> </ul>



Figura 3.59: PopUp Confirmacion

Es un pop up que se usará siempre que se requiera una confirmación ,se implementó para que pueda ser usado en cualquier contexto de la app, pasó las pruebas de aceptación planificadas.

### 3.9.2.1. Estado de la BD local

Cabe aclarar que la BD local es la memoria persistente de la app que solo existe en el dispositivo en el que esta instalada la app, y tiene una estrucura tipo

SQL.

Al finalizar esta iteración, el estado de la BD queda tal y como se muestra en la figura fig. 3.60

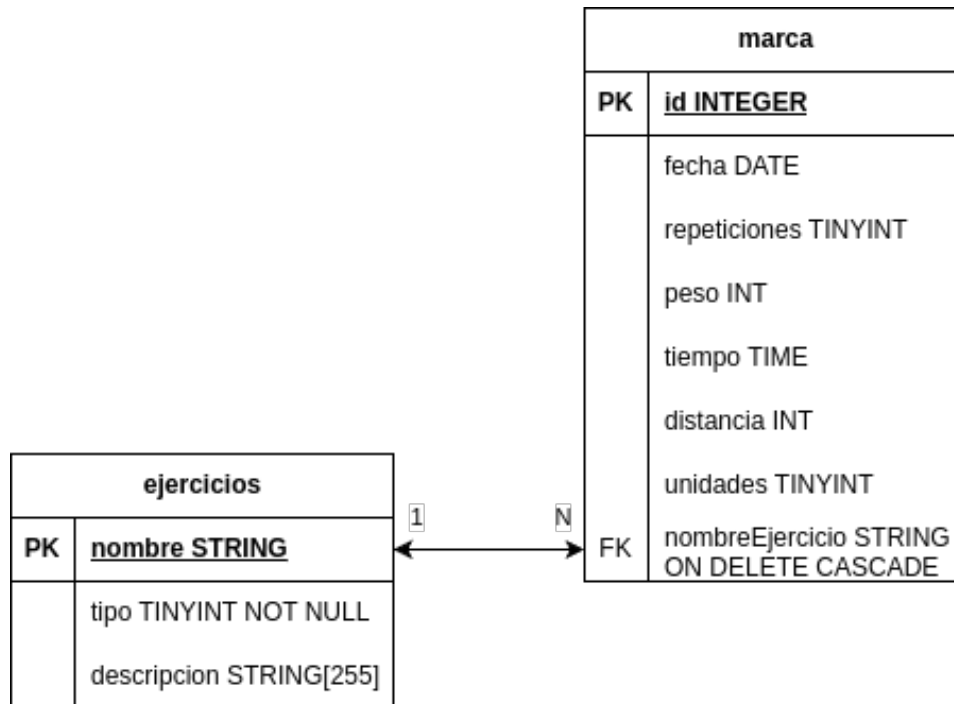


Figura 3.60: BD local iteracion 1

### 3.9.2.2. Sprint Review 1

Las primeras iteraciones tienden a ser más lentas debido a la fase inicial del desarrollo. No se completó la totalidad del sprint; quedó pendiente la subtask de modificar ejercicio en la base de datos (SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios). Aun así, las expectativas son positivas, ya que se espera un aumento en la velocidad de desarrollo.

La pantalla en la que se enseñan los datos de un ejercicio se modificó para facilitar su lectura y entendimiento, dado que su anterior diseño era confuso (ver figuras fig. 3.61 y fig. 3.62).

Otra parte a modificar es la del pop up de confirmación, ya que debido a la poca accesibilidad de los pop ups se sustituirá por una pantalla independiente.



Figura 3.61: Pantalla nueva

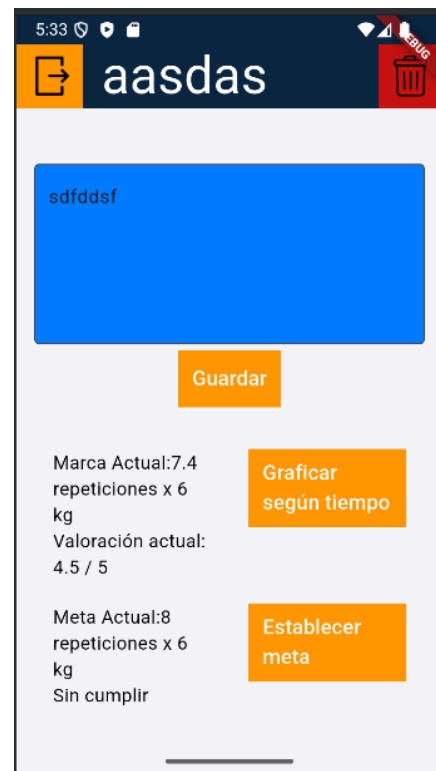


Figura 3.62: Pantalla antigua

También se cambiaron todos los pop ups y se sustituyeron por pantallas completas, se hizo esto para cumplir con las guías de accesibilidad dado que según la guía pueden llegar a confundir al usuario en la navegación. Estos son los pop ups sustituidos :

- Los pop ups de confirmacion
- Crear ejercicios
- Modificar ejercicios
- Nueva meta en un ejercicio
- Crear rutinas
- Modificar rutinas
- Modificar ejercicios de una rutina
- Información de una rutina a descargar
- Modificar informacion del perfil del usuario

Por consecuencia las futuras funcionalidades en la que se mencionan los pop ups, pasaran a ser pantallas completas.

### 3.9.3. Iteración 2

Esta iteración se centró en el desarrollo del sistema de sesiones, la API de la aplicación, el backend básico del servidor y la sección de rutinas. Las historias que incluye son las siguientes:

- SCRUM-18 Crear/Borrar mi usuario
- SCRUM-20 Iniciar/Cerrar sesión
- SCRUM-5 Insertar/Borrar/Modificar rutina
- SCRUM-4 Buscar rutina en la lista del usuario
- SCRUM-25 Implementar menú principal

También se terminará la historia no finalizada de la iteración anterior(SCRUM-3: Insertar/Borrar/Modificar ejercicio de la lista de ejercicios).

SCRUM-25 Implementar menú principal
Tareas:
1. Implementar el boceto IU fig. 3.6
2. <u>Añadir accesos a las funcionalidades actuales</u>
Pruebas de aceptación:
■ si la funcionalidad seleccionada no está implementada, no hacer nada
■ si la funcionalidad seleccionada está implementada, dar acceso a dicha funcionalidad



Figura 3.63: Menu Principal semifuncional

Pasó todas las pruebas de aceptación planificadas.

SCRUM-18 Crear/Borrar mi usuario
Tareas:
1. Implementar bocetos de IU fig. 3.4 ,fig. 3.5, fig. 3.2 y la parte necesaria de fig. 3.21
2. Implementar funciones de inserccion y borrado en el backend
Pruebas de aceptación:
<ul style="list-style-type: none"><li>■ si el usuario que se quiere crear ya existe, informar y pedir que se rellenen los credenciales otra vez</li><li>■ si el usuario que se quiere crear no existe, crear usuario y pasar a la siguiente pantalla</li><li>■ si cualquier dato requerido está en un formato incorrecto ,informar al usuario para que vuelva a rellenar estos</li><li>■ si todos los datos están en el formato correcto ,seguir con la creación del usuario</li><li>■ si se ha creado exitosamente el usuario, volver a la pantalla inicial</li><li>■ si se borra el usuario, borrar el token en el dispositivo y sus credenciales en el backend</li></ul>

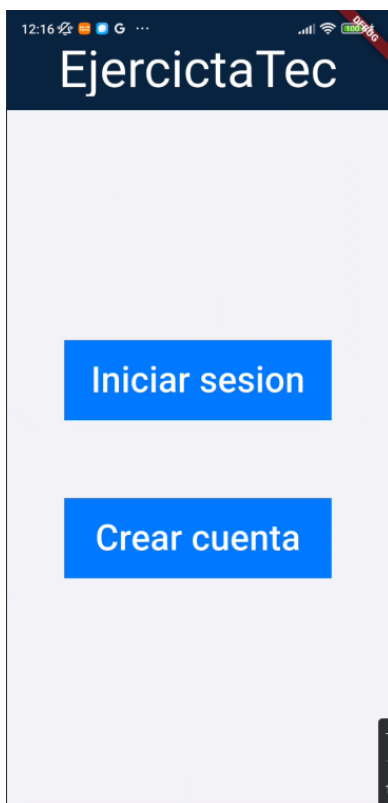


Figura 3.64: Log In / Sing In

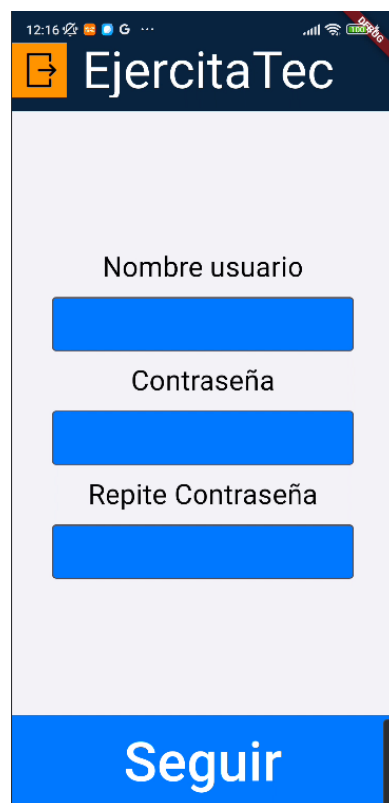


Figura 3.65: Sing In pantalla 1

The screenshot shows the 'Sing In' screen of the 'EjercitaTec' app. At the top, there's a status bar with the time 12:16 and various icons. Below it, the app's logo 'EjercitaTec' is displayed. The main content area is a light gray rectangle containing the following elements:

- Fecha Nacimiento**: A label above a blue button labeled 'Seleccionar' and the text 'No seleccionada'.
- Género**: A label above a dropdown menu showing 'Prefiero no decirlo' with a downward arrow.
- Peso en KG**: A label above a solid blue rectangular input field.
- Altura en cm**: A label above another solid blue rectangular input field.

At the bottom of the screen, there is a large blue button with the white text 'Crear usuario'.

Figura 3.66: Sing In pantalla 2

Cabe decir que siempre que se crea un usuario se crea un archivo en su dispositivo para almacenar su BD con el nombre del propio usuario. De esta forma se separa el almacenamiento local de los usuarios, por si el usuario quisiera compartir dispositivo con otros usuarios.

Pasó todas las pruebas de aceptación planificadas.



SCRUM-20 Iniciar/Cerrar sesión
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar el boceto IU fig. 3.3 y parte necesaria de fig. 3.21</li> <li>2. Implementar función de consulta en el backend</li> </ol>
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si el usuario con esa contraseña no existe, informar al usuario</li> <li>■ si el usuario con esa contraseña existe, devolver un token para su guardado y guardar usuario</li> <li>■ si el usuario quiere cerrar sesión, borrar token y nombre del usuario del dispositivo y volver a la pantalla inicial fig. 3.2</li> </ul>

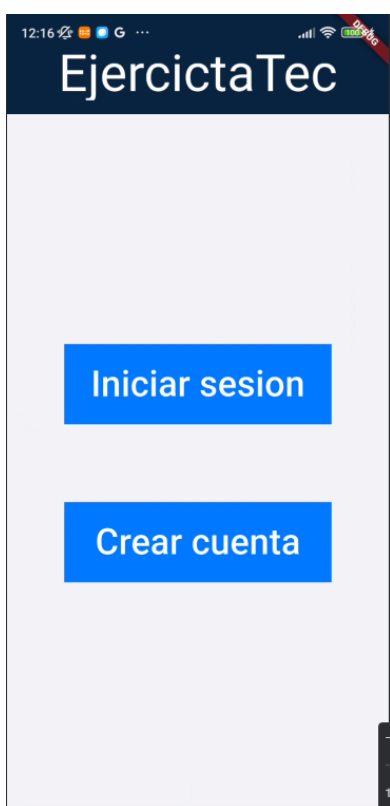


Figura 3.67: Log In / Sing In

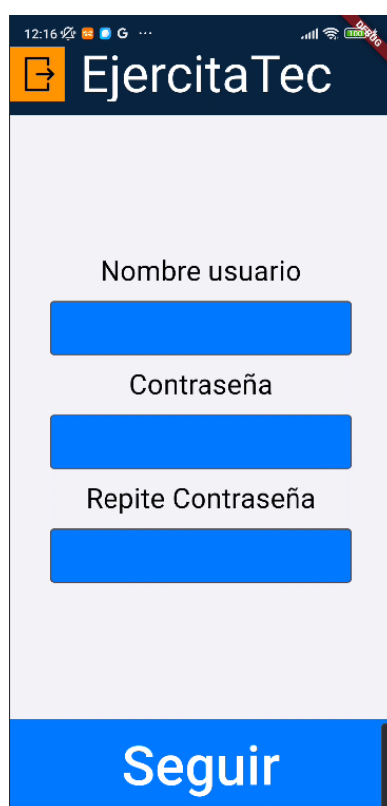


Figura 3.68: Sing In pantalla 1

Al hacer el login correctamente se otorga un token que no hay que renovar hasta el día siguiente, al cerrar sesión se borra automáticamente. Las pruebas realizadas se basaron en comprobar el correcto almacenaje y borrado del token, usando la librería de secure storage de flutter, que nos permite guardar variables de forma permanente y encriptada con un facil acceso, como si fuera un map.

Para comprobar la veracidad de los credenciales se implementó la parte básica

del backend, un server con una BD(mysql) que guardará los usuarios con sus contraseñas y la parte implementada usando Express.js(Node.js) para escuchar peticiones, el server también será el encargado de proporcionar y verificar los tokens. La comunicación entre servidor y cliente (el dispositivo en el que esta instalada la app) se usan peticiones HTTP, en futuro se puede migrar a HTTPS si es necesaria mas seguridad.

Se hicieron pruebas abriendo y cerrando sesión comprando si existía el token en memoria e intentando acceder a la app usando un token caducado. Siempre se añadía/borraba el token cuando se iniciaba/cerraba sesión y no dejaba entrar usando tokens caducados. Pasó todas las pruebas.

SCRUM-4 Buscar rutina en la lista del usuario
Tareas: <ol style="list-style-type: none"><li>1. Implementar IU del boceto fig. 3.13</li><li>2. Implementar lo necesario en la BD local</li><li>3. Implementar algoritmo de búsqueda</li><li>4. Implementar acceso a la rutina seleccionada</li></ol>
Pruebas de aceptación: <ul style="list-style-type: none"><li>■ si no existe la rutina buscada, no enseñar acceso en pantalla</li><li>■ si existe la rutina buscada, enseñar acceso en pantalla</li></ul>

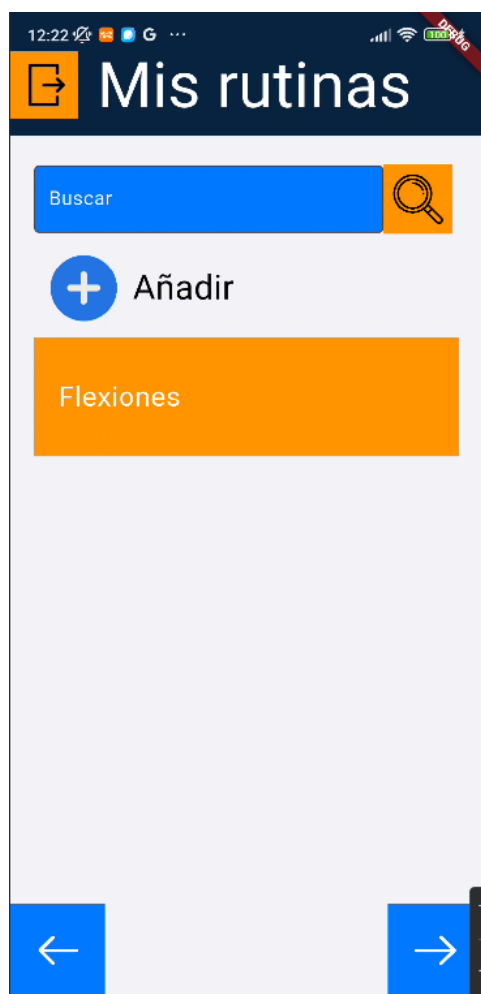


Figura 3.69: Lista Rutinas

En esta funcionalidad se aprovechó lo implementado en la iteración anterior del SCRUM-23. Se realizaron las mismas pruebas pero relacionadas con las rutinas. Como la funcionalidad previamente mencionada se diseñó para poder usar código a modo de plantilla y pasó todas las pruebas, esta como era de esperar las pasó también.

SCRUM-5 Insertar/Borrar/Modificar rutina
Tareas:
<ol style="list-style-type: none"><li>1. Implementar las insercciones en para la BD local</li><li>2. Implementar los borrados para la BD local</li><li>3. Implementar las modificaciones para la BD local</li></ol>
Pruebas de aceptación:
<ul style="list-style-type: none"><li>■ si se hace una insercción, que el elemento insertado sea visible en la lista de ejercicios</li><li>■ si se hace un borrado, que el elemento desaparezca de la lista</li><li>■ si se hace una modificación, que se hagan visibles los cambios al momento</li><li>■ si hay algún fallo en alguna de estas operaciones, informar al usuario con el motivo</li></ul>

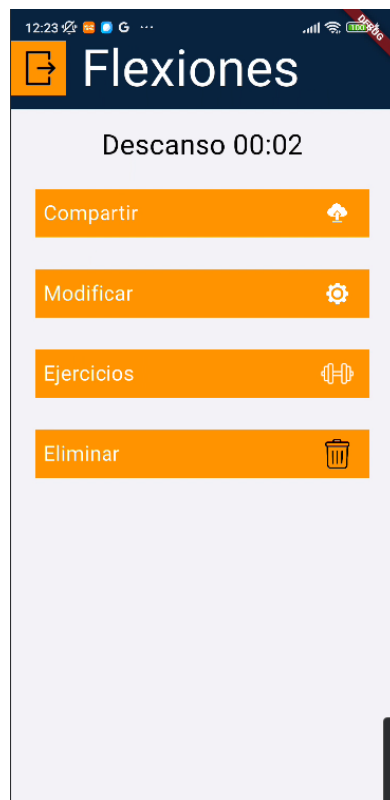


Figura 3.70: Datos Rutina

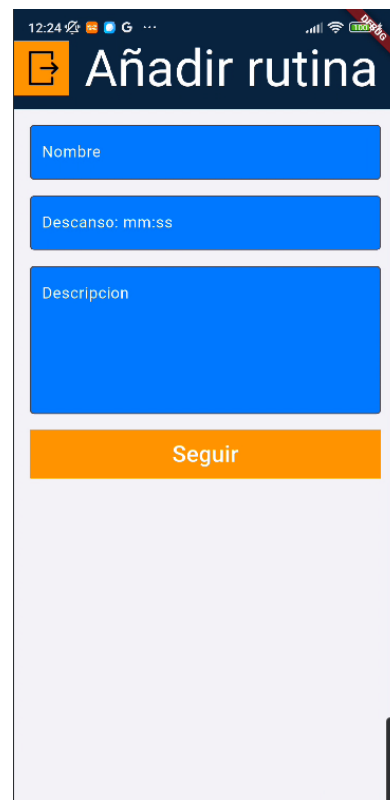


Figura 3.71: Crear Rutina

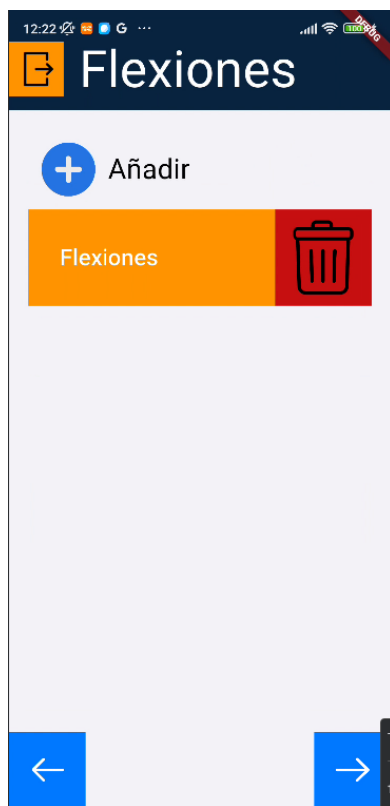


Figura 3.72: Lista Ejercicios Rutina

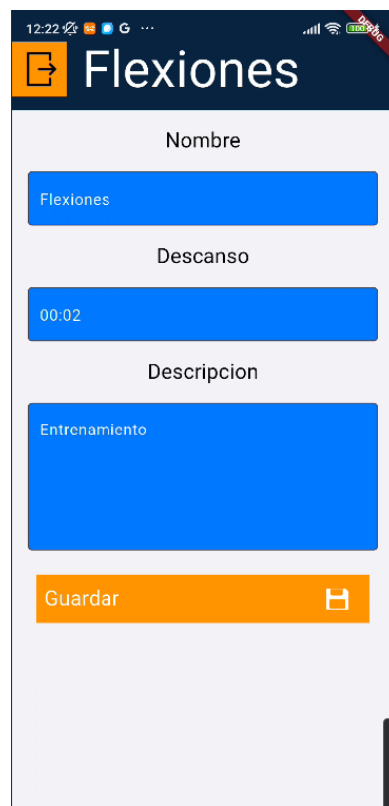


Figura 3.73: Modificar Rutina

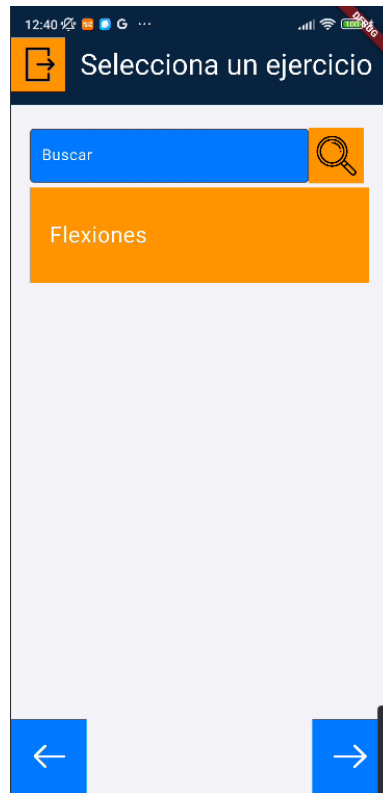


Figura 3.74: Lista añadir ejercicio a rutina

Gracias a que el código de la primera iteración se hizo muy reutilizable, solo hubo que hacer unas pequeñas modificaciones en la clase plantilla usada para crear las pantallas fig. 3.72 y fig. 3.74, en concreto a la pantalla de la lista de ejercicios perteneciente a la rutina se le realizaron pruebas para ver la correcta actualización de los contenidos.

En la pantalla fig. 3.73 la única prueba realizada fue comprobar la correcta modificación en la BD local y su correcta visualización en la misma.

La funcionalidad de compartir todavía no está implementada.

Se pasaron todas las pruebas de aceptación planificadas.

### 3.9.3.1. Sprint Review 2

Se sugirieron mejoras menores como el ajuste de tamaños e iconos en los botones de la sección de creación de rutinas.

También se modificó la funcionalidad de compartir rutinas. Ahora todas son modificables, eliminando la distinción entre rutinas descargadas (antes no modificables) y creadas (modificables). Esto mejora la experiencia del usuario: si desea

volver a una rutina original, simplemente la puede volver a buscar y descargar.  
Se cumplieron todas las funcionalidades en este sprint.

### 3.9.3.2. Estado de la BD local

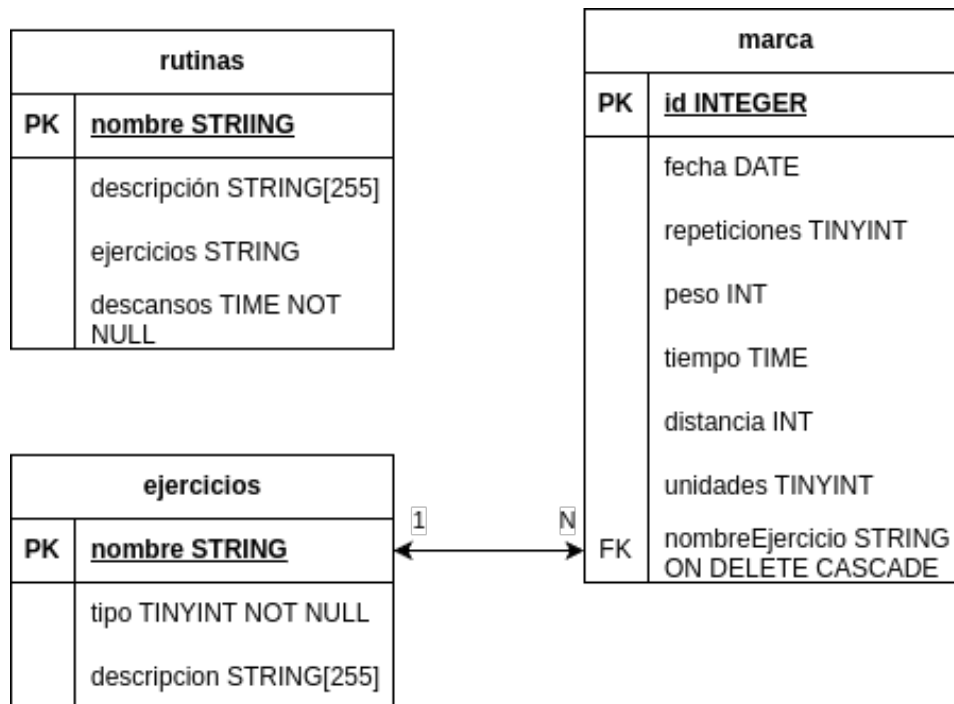


Figura 3.75: BD local iteracion 2

En la tabla rutinas la fila ejercicios es una cadena en la que se guardan los ejercicios que se van a realizar. No se usa una tabla índice porque se pueden repetir ejercicios en un orden determinado.

Se añadieron las tablas de ejercicios y marcas, una o varias marcas siempre están relacionadas con un ejercicio, de esta forma, se saben todas las marcas de un ejercicio y si se elimina un ejercicio sus marcas también desaparecen.

### 3.9.3.3. Estado de la BD del servidor

En esta iteración se comienza a desarrollar la BD del servidor que es donde se guardan los usuarios registrados y sus rutinas compartidas. De esta forma la diferenciamos de la BD local, que es el almacenamiento interno del dispositivo.

usuario	
PK	<u>username VARCHAR[50]</u>
	passwd VARCHAR[255]

Figura 3.76: BD servidor iteracion 2

### 3.9.4. Iteración 3

Esta iteración se centró en desarrollar la funcionalidad para compartir rutinas entre usuarios, incluyendo la subida, visualización y descarga de rutinas. Además, se comenzó a implementar la funcionalidad de resumen de datos para optimizar el uso de la memoria local. Incluye las siguientes historias:

- SCRUM-2 Establecer peso objetivo
- SCRUM-29 Copiar rutina
- SCRUM-8 Enseñar datos de una rutina a descargar
- SCRUM-9 Compartir mi rutina
- SCRUM-27 Buscar rutinas para descargar
- SCRUM-7 Revisar datos para ver si el descanso es necesario
- SCRUM-19 Resumir datos

SCRUM-2 Establecer peso objetivo
Tareas:
<ol style="list-style-type: none"> <li>1. Implementar la parte necesaria de fig. 3.21</li> <li>2. Implementar el almacenaje de ese peso objetivo</li> </ol>
Pruebas de aceptación:
<ul style="list-style-type: none"> <li>■ si el peso objetivo está en el formato correcto, guardar</li> <li>■ si el peso objetivo no está en el formato correcto, enseñar una alerta para su corrección</li> </ul>



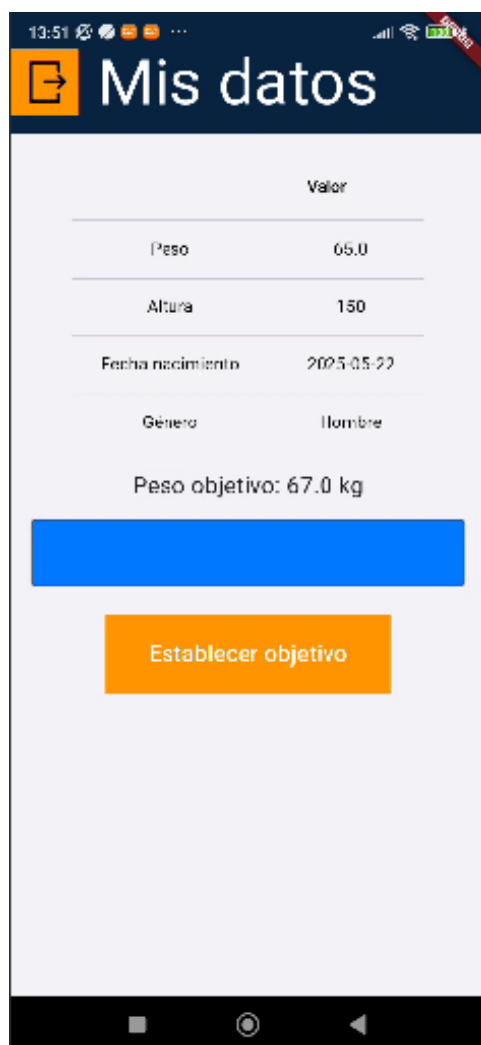


Figura 3.77: Peso Objetivo

Para guardar el peso, en esta iteración como no se han creado aún las tablas para guardar los pesajes del usuario se guardará usando secure storage, es una solución temporal. Para esta funcionalidad se hicieron pruebas para comprobar la correcta insercción del formato del peso, que no se añadan letras. que se admitan comas y puntos, que solo se use un decimal. Pasó todas las pruebas.

SCRUM-9 Compartir mi rutina
Tareas: <ol style="list-style-type: none"><li>1. Implementar la BD en el backend</li><li>2. Implementar funciones de inserción en la API</li></ol>
Pruebas de aceptación: <ul style="list-style-type: none"><li>■ si la rutina no contiene ejercicios, no permitir su subida al servidor</li><li>■ si la rutina contiene ejercicios, permitir la subida al servidor</li></ul>

Esta funcionalidad sirve para subir la rutina a la BD del servidor para que este disponible para su descarga.

Dió muchos quebraderos de cabeza ya que varios usuarios podrían tener una rutina con el mismo nombre y al mismo tiempo al descargar una rutina, un usuario podría tener un ejercicio llamado igual o varios usuarios podrían tener en la nube varios ejercicios subidos con el mismo nombre. Estos problemas se detectaron realizando pruebas y para solucionarlos se optó por no usar el nombre de rutina o ejercicio como claves primarias, por lo que se hizo lo siguiente: en la BD del servidor, los ejercicios y rutinas serán referenciadas usando un entero autoincrementado como id.

El resto de pruebas se realizaron de forma satisfactoria.

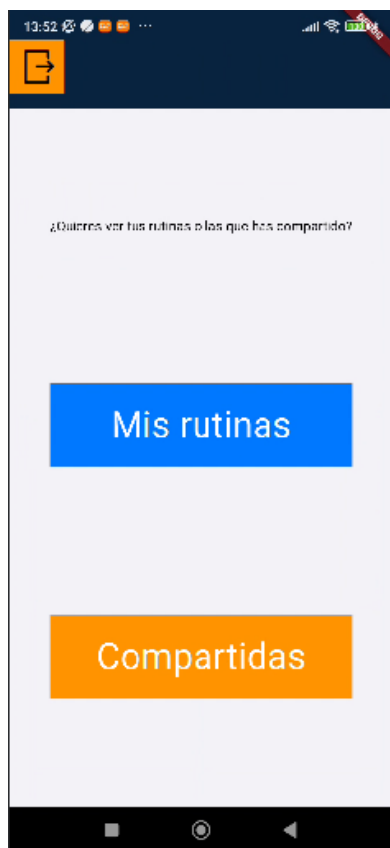


Figura 3.78: Elegir lista compartida por el usuario o locales

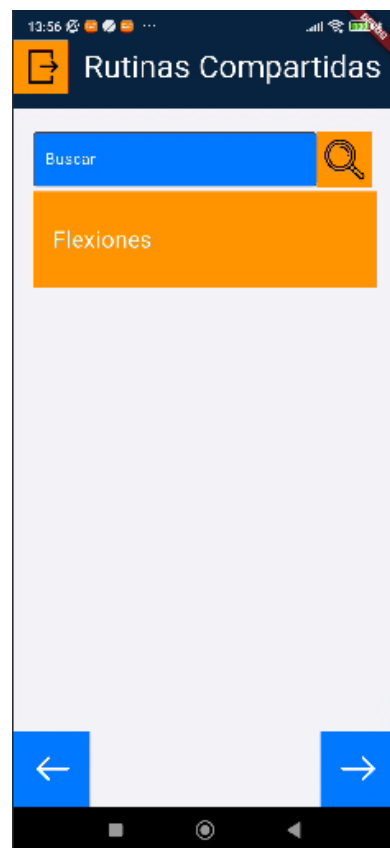


Figura 3.79: Rutinas compartidas por el usuario

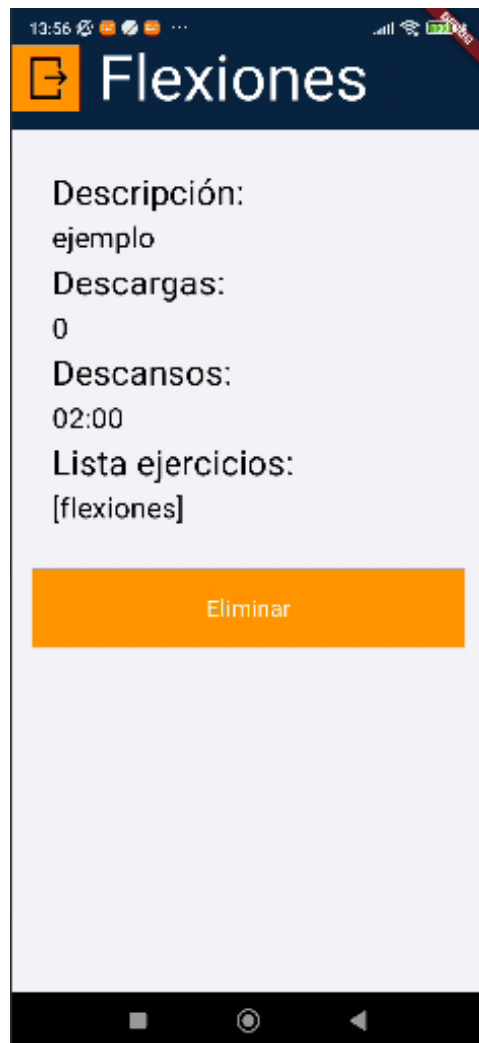


Figura 3.80: Datos rutinas compartidas por el usuario

SCRUM-27 Buscar rutinas para descargar
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar funciones de consultas en la API</li> <li>2. Implementar IU de búsqueda de rutinas por usuario creador y por nombre de rutina</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si busco un usuario que no existe, no se muestra ningún acceso en la IU</li> <li>■ si busco un usuario existente, se muestra su acceso en la IU</li> <li>■ si busco una rutina que no existe, no se muestra ningún acceso en la IU</li> <li>■ si busco una rutina existente, se muestra su acceso en la IU</li> </ul>

Se puede buscar tanto un usuario y ver las rutinas que ha subido o directa-

mente buscar por nombre de la rutina. Se recicló la pantalla de lista de rutinas fig. 3.13, y se implementó la pantalla de DatosRutinaComp, que se usa para ver datos de pantallas compartidas. También para hacer más fácil la distinción entre rutinas del propio usuario y las que ha compartido se añadió otra pantalla exclusivamente de pantallas compartidas, en la cual se nos permite eliminar de la nube las rutinas elegidas.

Haciendo pruebas descargando varias veces la misma rutina y casos distintos fue fácil detectar algunos problemas de integridad debido a la repetición de nombres. Al descargar una rutina se le añadirá a su nombre y al de los ejercicios que contiene el nombre del creador, y si aún así hay problemas de integridad, se añade al nombre de la rutina y/o ejercicio el número de copia que es dentro del dispositivo. Desarrollando lo anterior, al descargar una rutina llamada 'Flexiones' del usuario 'Paco', en el dispositivo que realiza la descarga se llamará 'Flexiones-Paco', si en el dispositivo en el que se realiza la descarga ya existe otra rutina llamada 'Flexiones-Paco', la descargada se llamará en el dispositivo 'Flexiones-Paco(1)'. La integridad de los ejercicios se trata igual.

Una vez implementadas estas soluciones, ya se pasaron exitosamente las pruebas.

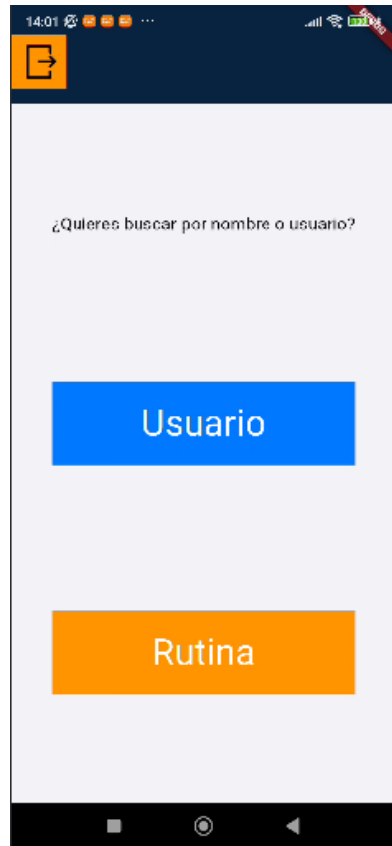


Figura 3.81: Buscar rutinas o usuario

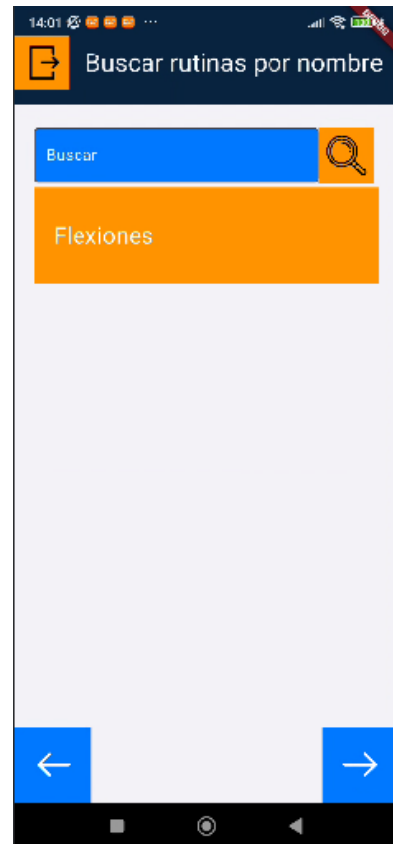


Figura 3.82: Buscar rutinas por su nombre

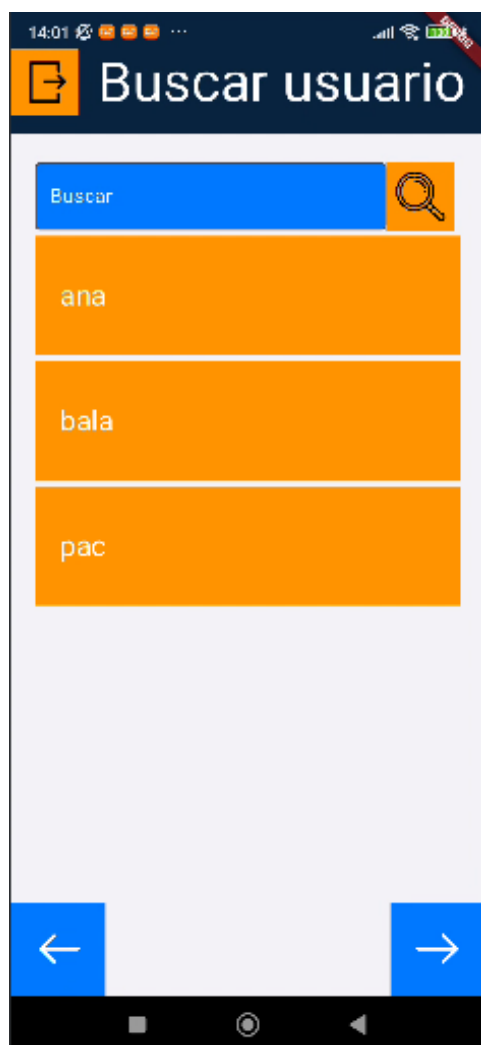


Figura 3.83: Buscar usuarios

SCRUM-8 Enseñar datos de una rutina a descargar
Tareas: <ol style="list-style-type: none"><li>1. Implementar la IU</li><li>2. Implementar las funciones de consulta en la API</li><li>3. Implementar las funciones de inserción en la BD local</li></ol>
Pruebas de aceptación: <ul style="list-style-type: none"><li>■ si se va a descargar la rutina y el usuario tiene en el dispositivo otra con el mismo nombre, un algoritmo resolverá este problema</li><li>■ si se va a descargar la rutina y el usuario no tiene en el dispositivo otra con el mismo nombre, descargar normalmente</li></ul>

Se recicló una de las pantallas de SCRUM-9 (fig. 3.80), solo cambió el botón que se usa para descargar. Se hicieron pruebas comprobando que solo podía borrar la rutina el usuario creador y que los datos enseñados al descargar la rutina eran correctos.

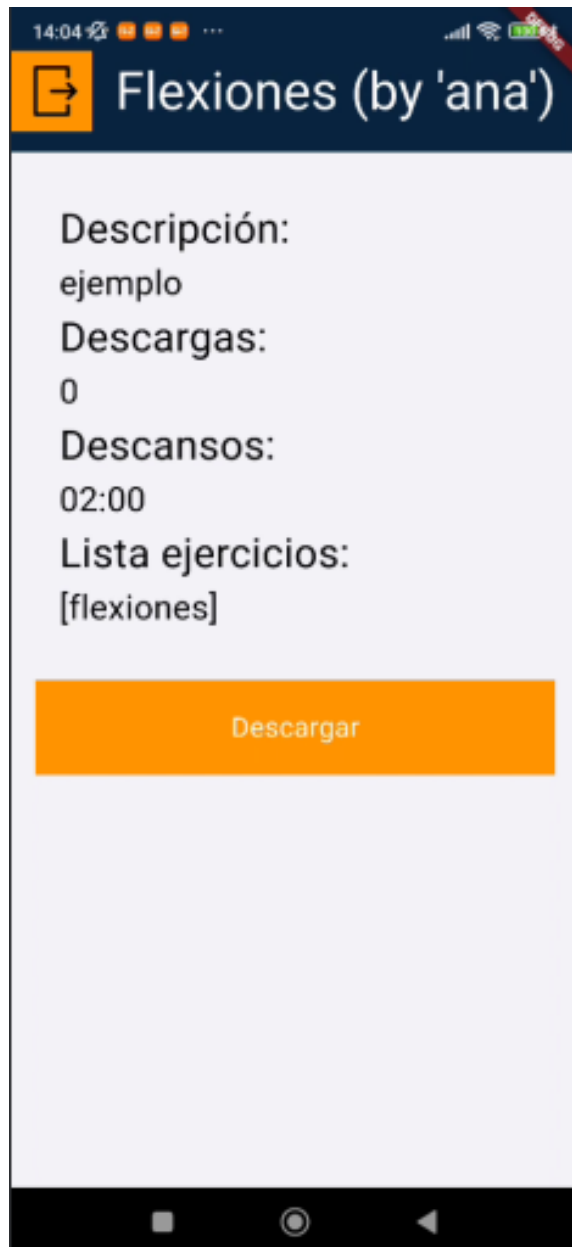


Figura 3.84: Datos rutinas para descargar



SCRUM-29 Copiar rutina
Tareas: <ol style="list-style-type: none"><li>1. Añadir al boceto IU de datos rutina fig. 3.15 esta funcionalidad</li><li>2. Implementar la IU resultante de la tarea anterior</li><li>3. Implementar las consultas e inserciones necesarias en la BD local</li></ol>
Pruebas de aceptación: <ul style="list-style-type: none"><li>■ si hay algún problema por coincidencia de nombres, un algoritmo resolverá este problema</li><li>■ si no hay algún problema por coincidencia de nombres, copiar</li></ul>

#### 3.9.4.1. Funcionalidad descartada

Durante la realización de estas tareas de usuario, se dió notoriedad a que las historias restantes de este sprint (SCRUM-7 Revisar datos para ver si el descanso es necesario y SCRUM-19 Resumir datos) no pueden ser implementadas en este punto del desarrollo. Se pospondrán para su futura implementación.

Se ha decidido dar menos prioridad y que por tanto no se abordará en este TFG por limitaciones de tiempo en concreto el SCRUM-7 (Revisar datos para ver si el descanso es necesario), porque necesitaría los datos de un smartwatch y ha sido una funcionalidad previamente descartada.

También se ha reducido la prioridad de SCRUM-29 (Copiar rutina), dado que aporta poco valor a la app.

#### 3.9.4.2. Estado BD local

No ha habido cambios respecto a la iteración anterior

### 3.9.4.3. Estado BD del servidor

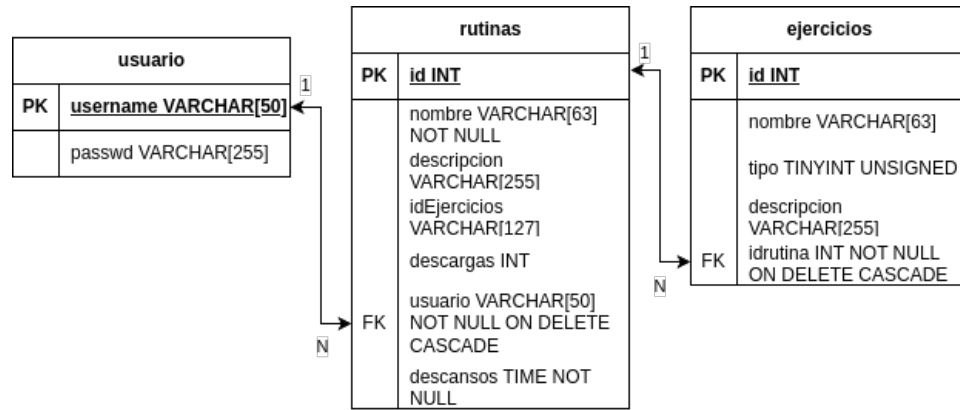


Figura 3.85: BD del servidor iteracion 3

### 3.9.4.4. Sprint Review 3

Se identificaron dos errores principales de planificación:

**Error 1:** Se planificó implementar la funcionalidad de resumir datos sin haber completado la obtención de datos.

**Solución:** Replanificar los sprints.

Los imprevistos y problemas surgidos durante el desarrollo han permitido detectar *bugs*, errores de diseño y carencias funcionales que de otro modo podrían haber pasado desapercibidos. Gracias a ello, se han podido proponer nuevas funcionalidades y mejorar las ya existentes, lo cual contribuye significativamente a mejorar la calidad global de la aplicación.

Algunas funcionalidades propuestas como mejora para la siguiente iteración son:

- Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Al seleccionar la opción de borrar usuario del dispositivo, eliminar también la base de datos local asociada.
- Crear una base de datos local independiente para cada nuevo usuario creado en un dispositivo.
- Posibilidad de editar el nombre de una rutina.
- Marcar los ejercicios eliminados con una *flag*, para evitar que se inicien rutinas que los contengan.
- Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Estas funcionalidades están pensadas para aportar mayor **seguridad, integridad y calidad** al producto final.

En esta review se estuvo debatiendo si quitar las funcionalidades relacionadas con el smartwatch, por cuestiones de tiempo, ya que con las funcionalidades añadidas en el capítulo anterior se va ajustando el plazo.

En esta iteración se tenía pensado implementar la funcionalidad de resumir los datos, no obstante, al no tener implementada la parte en la que se guardan las marcas de los entrenamientos, no es posible trabajar en esa funcionalidad. Por lo tanto, se ha pospuesto para la próxima iteración. También se le dió el visto bueno a la implementación de las funcionalidades previamente expuestas durante el desarrollo.

### 3.9.5. Iteración 4

En esta iteración se tiene pensado implementar la funcionalidad del calendario, donde el usuario podrá ver su planificación y acceder a sus marcas. Por otro lado, también se implementarán las funcionalidades nuevas del sprint anterior y las nuevas pantallas.

- SCRUM-26: Detectar metas cumplidas en los ejercicios después del entrenamiento
- SCRUM-1: Registrar peso por día
- SCRUM-16: Realizar el flujo del entrenamiento
- SCRUM-28: Implementar calendario

SCRUM-28: Implementar calendario
<p>Tareas:</p> <ol style="list-style-type: none"><li>1. Implementar la IU e insertarla en el menu principal fig. 3.6</li><li>2. Implementar en la BD local</li><li>3. Implementar funciones de inserción en BD local</li><li>4. Implementar funciones de consulta en BD local</li><li>5. Implementar funciones de borrado en BD local</li><li>6. Implementar un acceso para ver los datos registrados sobre un día en concreto</li></ol>
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"><li>■ si un día no tiene una rutina asignada, se considera descanso</li><li>■ si un día tiene una rutina asignada, no hay marcas registradas y el día ya ha pasado, no se enseñará información, ya que se considera no entrenado</li><li>■ si un día tiene una rutina asignada, no hay marcas registradas pero es hoy, se enseñará un acceso para empezar el entrenamiento, se considera que el usuario todavía no ha entrenado</li><li>■ si un día tiene una rutina asignada, pero es futuro, no se enseñará un acceso para empezar el entrenamiento</li><li>■ si un día tiene una rutina asignada y hay marcas registradas sea el día actual o pasado, se mostrará información de las marcas, sin acceso a entrenamiento</li><li>■ si se cambia la rutina a entrenar, se inserta en la BD del dispositivo y se hace visible en el calendario</li><li>■ si se cambia la rutina asignada en un día por un descanso, se borra de la BD y se hace visible en el calendario</li></ul>

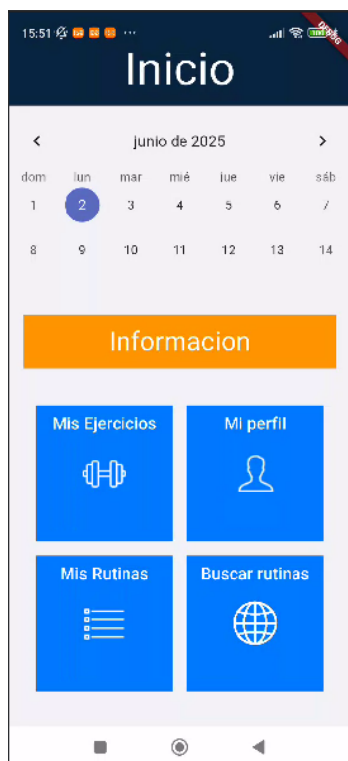


Figura 3.86: Calendario

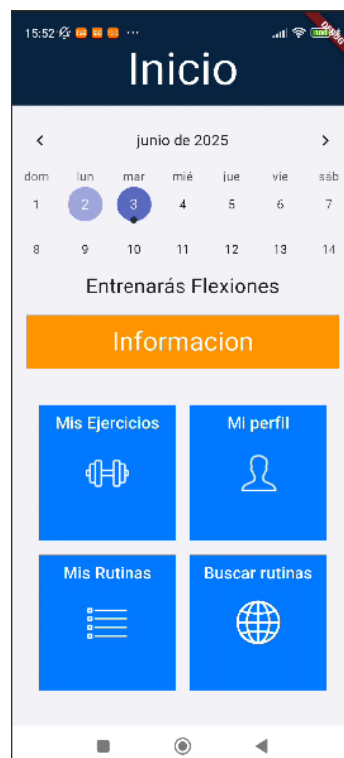


Figura 3.87: Calendario con eventos

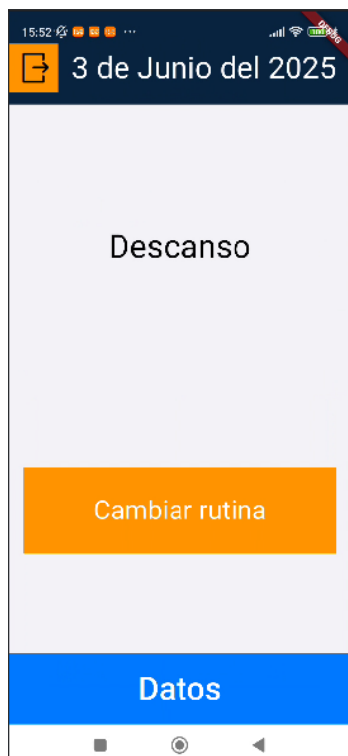


Figura 3.88: Descanso

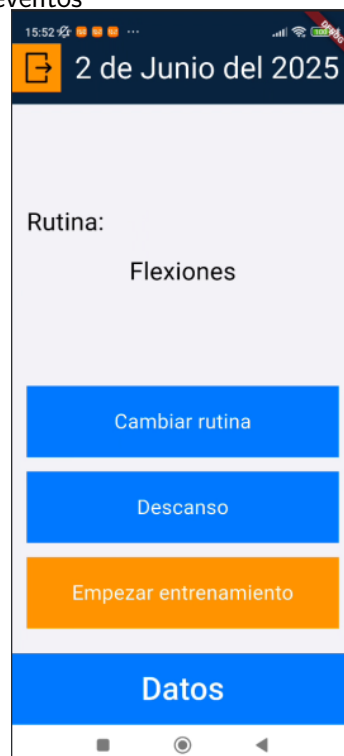


Figura 3.89: Entrenar

El calendario es donde el usuario podrá organizar sus entrenamientos, acceder a sus marcas y datos sobre su peso guardados por días. El calendario está implementado usando la librería de tableCalendar, que permite trabajar con calendarios con una alta personalización de la estética y trabajar comodamente con los datos reflejados en el mismo usando el tipo DateTime de flutter. Para guardar eventos se almacenan en una tabla en la BD local llamada entrenamientos. Cada entrenamiento se guarda usando la fecha de cuando se realizó/realizará como primary key junto con la rutina que se hizo o planea hacerse. Como no se permite modificar las rutinas una vez ya se haya entrenado o pasado el día, el único problema de integridad es cuando se añade un evento a un día ya asignado, lo cuál resolvemos sustituyendo el nuevo por el antiguo. Las marcas funcionan de manera similar, primary key es un int auto incrementado, pero además cada marca tiene asociada una fecha. Importante, cada marca se asocia a los resultados de una serie.

Se realizaron pruebas comprobando si se actualizaban los entrenamientos y pesajes añadidos y se veían reflejados correctamente en el calendario. Todos los datos se guardan correctamente así que pasó las pruebas de aceptación.

SCRUM-16 Realizar el flujo del entrenamiento
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar IU de las pantallas de entrenamientos (de la fig. 3.33 a la fig. 3.40)</li> <li>2. Implementar una clase que controle el cambio entre distintas pantallas y el guardado de marcas</li> </ol>
<p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si la rutina asignada para un día, no tiene ejercicios y se desea iniciar el entrenamiento, se muestra por pantalla una alerta y no se permite continuar con el entrenamiento</li> <li>■ si la rutina asignada para un día, tiene ejercicios, comienza el recorrido entre IUs</li> <li>■ si es la primera serie de un ejercicio, no se permite terminar el ejercicio</li> <li>■ si no es la primera serie de un ejercicio, se permite terminar el ejercicio</li> <li>■ si acaba una serie, no se permitirá acabar el ejercicio o continuar con este hasta que termine el tiempo de descanso que se muestra en pantalla</li> <li>■ si finaliza el último ejercicio de la lista, fin del entrenamiento, se devuelve al usuario al menú principal y guardan las marcas obtenidas</li> </ul>

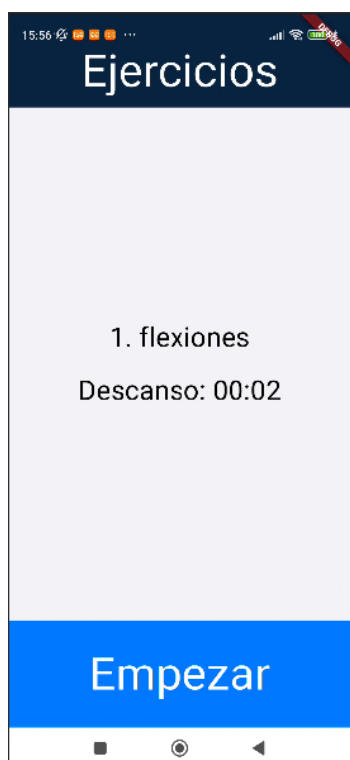


Figura 3.90: Lista inicial de ejercicios

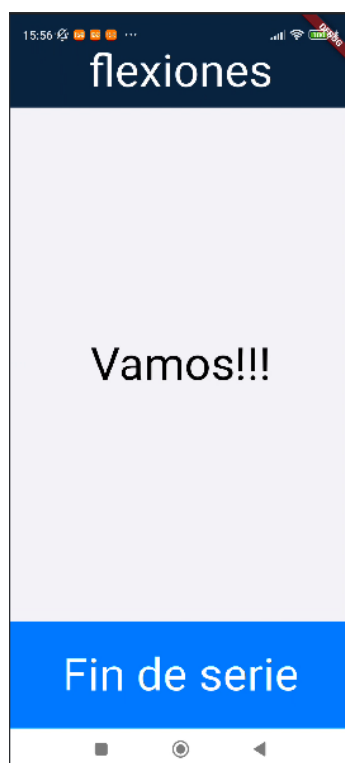


Figura 3.91: Realizando serie

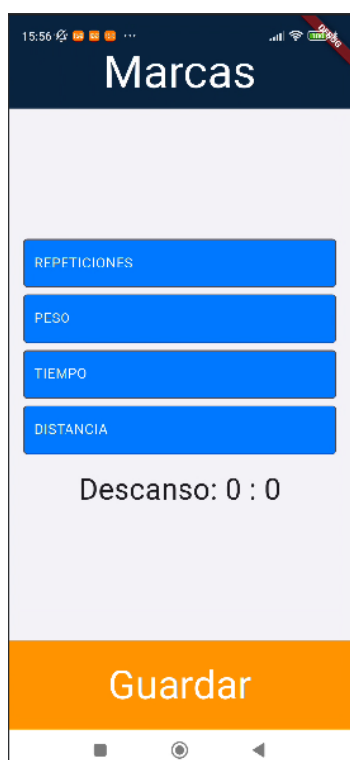


Figura 3.92: Guardando marcas

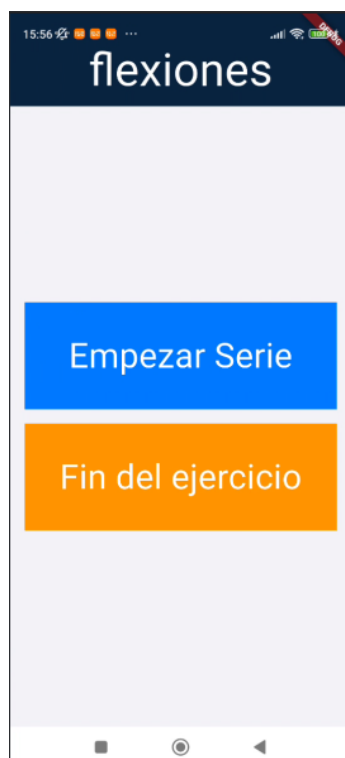


Figura 3.93: Pantalla enter series

Para abordar esta funcionalidad, se ha implementado una clase aparte encargada del desarrollo de esta parte de la app. Se ha decidido así ya que es una lógica más compleja y es conveniente tenerla separada de la IU.

Esta clase solo se puede crear de forma util llamando a un método estático de la propia clase (se hace así ya que necesita hacer operaciones asíncronas para obtener los datos necesarios), este método estático devuelve la instancia con la que vamos a trabajar. Se llama al método ejecutar de la instancia creada y sola se encarga de mostrar las pantallas al usuario, guardar los datos y de comparar las metas que se propuso el usuario.

Aquí un diagrama de actividades que explica la tarea de esta clase:

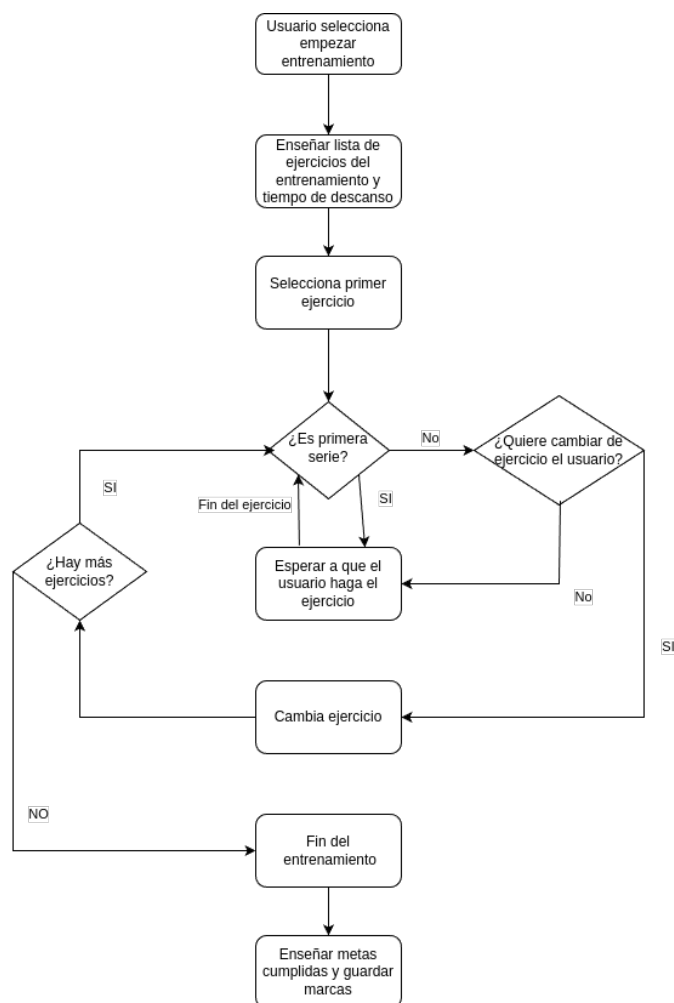


Figura 3.94: Flujo entrenamiento

Se hicieron pruebas simulando el flujo varias veces de distintas formas, com-



probando su correcta ejecución y que se guardaban correctamente los datos.  
Pasó todas las pruebas.

SCRUM-26 Detectar metas cumplidas en los ejercicios despues del entrenamiento
Tareas:
<ol style="list-style-type: none"><li>1. Implementar bocetos de IU fig. 3.41</li><li>2. Implementar lógica para comparar marcas y metas despues de acabar el entrenamiento en la clase que controla el flujo del entrenamiento</li></ol>
Pruebas de aceptación:
<ul style="list-style-type: none"><li>■ si en un ejercicio no se ha cumplido la meta, no se enseña nada</li><li>■ si en un ejercicio cumplimos una meta, se enseña en que ejercicio se cumplió la meta</li></ul>

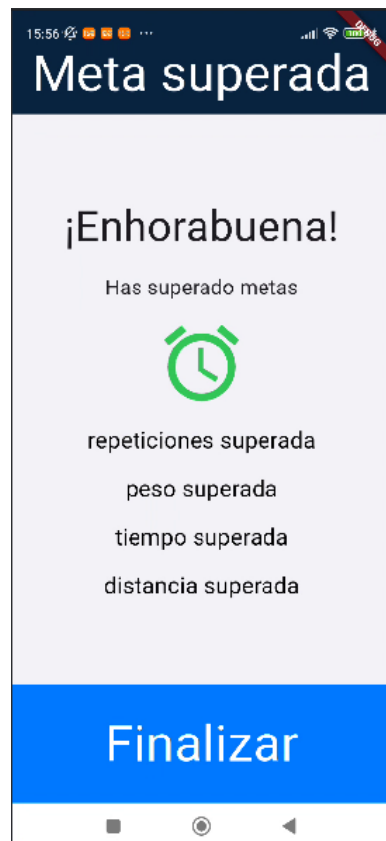


Figura 3.95: Meta superada

Esta funcionalidad es un añadido a la clase FlujoEntrenamiento, cuando se detecta el fin antes de subir las marcas, se ordenan por ejercicio y se comprueba si en algún momento se superó alguna meta y cual parámetro medido fue el superado. Se hicieron pruebas comprobando si de verdad se cumplían las metas

cuando saltaba el aviso en pantalla, pasó todas las pruebas.

SCRUM-1 Registrar peso por día
<p>Tareas:</p> <ol style="list-style-type: none"> <li>1. Implementar boceto de IU fig. 3.44</li> <li>2. Implementar en BD local</li> <li>3. Implementar funciones de inserción</li> <li>4. Implementar funciones de consulta</li> <li>5. Implementar lógica para determinar si un usuario quiere subir o bajar de peso</li> </ol> <p>Pruebas de aceptación:</p> <ul style="list-style-type: none"> <li>■ si se introduce algún dato como formato incorrecto, se muestra una alerta por pantalla</li> <li>■ si se introducen los datos en formato correcto, se guarda</li> </ul>

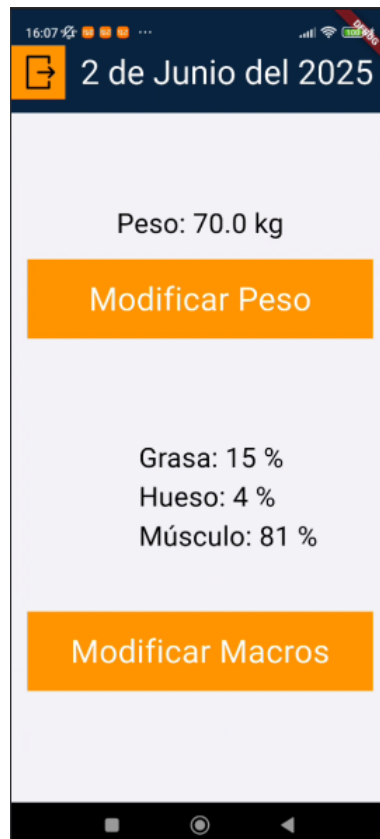


Figura 3.96: Registrar peso por día

Para implementar esta funcionalidad se añadió un acceso en la parte de abajo de la pantalla para cada día del calendario, dando igual si existe algún evento o no, siempre se puede guardar/modificar el peso del usuario siempre que no sea de un día pasado. Se realizaron pruebas comprobando los correctos formatos

al introducir el peso y los porcentajes, también se comprobó que se hacía un correcto display de estos datos al acceder a ellos desde el calendario. Pasó todas las pruebas.

En esta iteración hay pocas funcionalidades comparadas con otras, porque el flujo del entrenamiento es una funcionalidad que requirió más esfuerzo. Aparte se añade otra funcionalidad realizada en esta iteración que es la de la HU SCRUM-28.

### 3.9.5.1. Cambios en el backend

Los cambios en el backend están relacionados con las siguientes funcionalidades propuestas en la iteración anterior:

- Cambio 1: Verificar el token del usuario antes de permitir la subida de rutinas, para evitar suplantación.
- Cambio 2: Marcar a los usuarios permanentemente eliminados con una *flag*, para proceder a eliminarlos en los dispositivos.

Cambio 1: la solución implementada es la siguiente, a todas las funciones que puedan ser potencialmente víctimas de una suplantación (vía petición normal de http), se les ha añadido un middleware, una función que se ejecuta antes para verificar el token. Express.js permite implementar esto rápidamente

Cambio 2: de primeras se pensó en usar una *flag*, pero finalmente para evitar pérdida de rendimiento al momento de que el usuario realice un entrenamiento (que es cuando en un principio se tenía pensado borrar el ejercicio), se optó por modificar la lista de ejercicios de la rutina al momento de borrar el ejercicio. El número de consultas iba a ser el mismo, pero se ahorra memoria, ya que no añadimos una columna más a la tabla de ejercicios

Otro cambio en el backend que no tiene que ver con lo propuesto con la iteración anterior, es la adición de 2 campos a la tabla usuarios de la BD del servidor. Se añadió el campo *int* descargas, siendo este el número total de descargas que posee en sus rutinas el usuario en cuestión, y el campo *fechaCreacion* que es la fecha en la que se creó la cuenta en cuestión. El número total de descargas se usa para que cuando se busquen usuarios para descargar rutinas salgan los que más descargas poseen primero y la fecha de creación para que cuando la app se inicie cree un calendario que vaya desde la fecha de creación hasta 1 mes en adelante de la fecha actual, porque no tiene sentido añadir fechas de antes de crear la cuenta.

## 3.9.5.2. Estado de la BD local

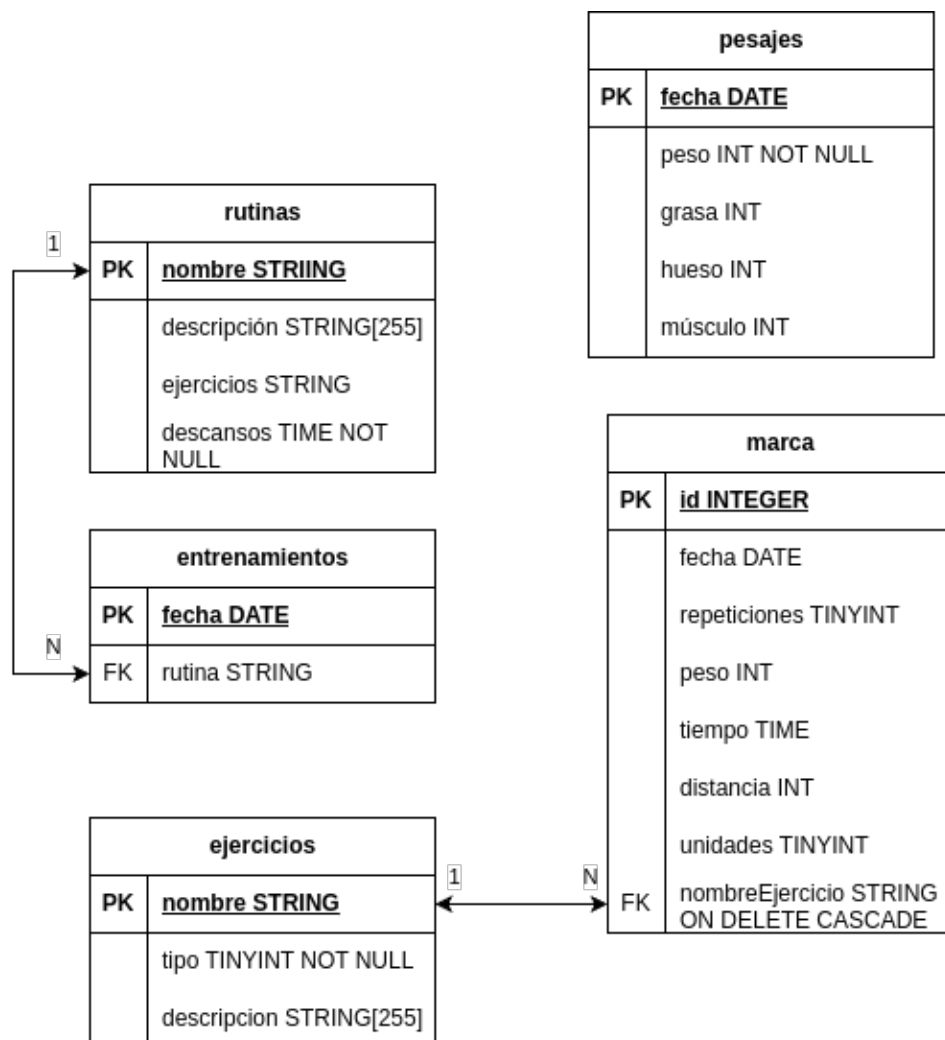


Figura 3.97: BD local iteración 4

### 3.9.5.3. Estado BD del servidor

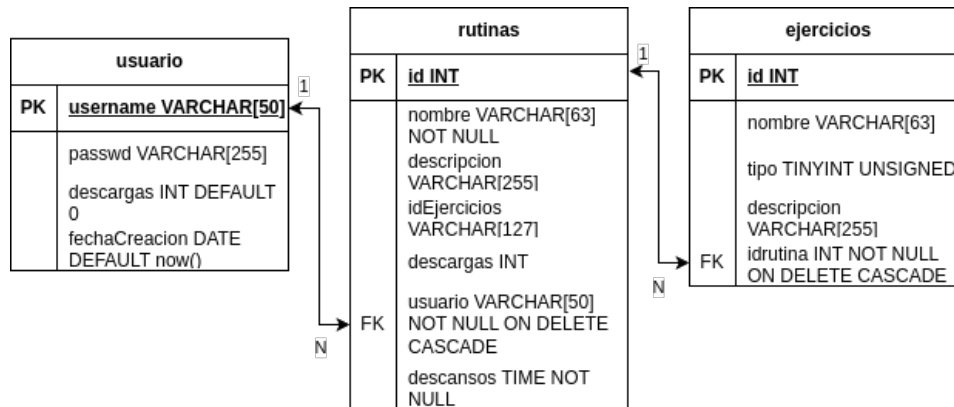


Figura 3.98: BD del servidor iteración 4

### 3.9.5.4. Sprint review 4

En esta review se ha propuesto que después de los entrenamientos se les enseñen a los usuarios las metas que tenían establecidas siempre, aunque no las hayan superado, evitando siempre el feedback negativo al usuario. Se tendrá en cuenta para la próxima iteración.

### 3.9.6. Iteración 5

Al principio de la iteración se dio a notar que faltaba por implementar la IU y algunas funciones para visualizar las marcas obtenidas por el usuario, lo cual se añadirá a esta iteración. Teniendo en cuenta el tiempo que queda, la funcionalidad de la IA no se podrá implementar. Si se dará prioridad a la funcionalidad de hacer gráficas usando las marcas obtenidas por el usuario, lo cual no se añadió desde un principio porque era necesario poder añadir marcas.

- SCRUM-34: Enseñar las marcas obtenidas en un día
- SCRUM-6: Hacer gráfica en base a las marcas obtenidas

SCRUM-34: Enseñar las marcas obtenidas en un día
Tareas: <ol style="list-style-type: none"> <li>1. Implementar las consultas en la BD local</li> <li>2. Implementar la IU</li> </ol>
Pruebas de aceptación: <ul style="list-style-type: none"> <li>■ si existen varios ejercicios separarlos en distintas pantallas</li> <li>■ pulsar en la pantalla para ir pasando series del mismo ejercicio</li> </ul>

Se añadió una nueva pantalla MarcasEntrenamientos, en la cual se muestran los resultados del entrenamiento de un usuario en sus distintas series. Pasó todas las pruebas.

SCRUM-6: Hacer gráfica en base a las marcas obtenidas
Tareas: <ol style="list-style-type: none"> <li>1. Implementar las consultas necesarias</li> <li>2. Implementar la IU</li> </ol>
Pruebas de aceptación: <ul style="list-style-type: none"> <li>■ hacer un display correcto de los distintos datos</li> </ul>

La grafica se implementó usando la librería `fl_chart` de flutter, la cual permite usar componentes gráficos para hacer distintos tipos de gráficas. Para el parámetro tiempo se representa el número total de segundos en el eje Y, para el resto de parámetros se usa su valor numérico. El eje X es la fecha de cuando se alcanzó la marca.

Cabe decir que la marca obtenida es la media del último entrenamiento realizado, dicha media la realiza la BD mediante una consulta.

Para esta funcionalidad es necesario hacer más pruebas, no se realizan más por falta de tiempo.

# Capítulo 4

## Conclusiones

### 4.1. Objetivos

Los objetivos perseguidos en este trabajo han sido los previamente mencionados en la sección objetivos en la introducción, los cuales son:

- Analizar algunas apps del mercado, así como sus características, qué ofrecen, su costo para el usuario y las valoraciones de los usuarios finales, para aclarar qué es lo que buscan los usuarios en este tipo de apps y considerar esas necesidades en el software a desarrollar.
- Investigar sobre qué herramientas usar para la implementación de la base de datos, backend, frontend, y técnicas y metodologías para dar un desarrollo de calidad al software y para ayudar a la sostenibilidad del sistema en el tiempo.
- Conocer y aprender a usar herramientas actuales y punteras para el desarrollo de aplicaciones móviles.
- Desarrollar una aplicación multiplataforma que cubra todos los requisitos necesarios para dar soporte a la planificación y realización de entrenamientos y monitorización de rutinas de ejercicio físico.
- Tratar de obtener un software lo más accesible posible.

Analizando cada objetivo individualmente:

1. **Analizar algunas apps del mercado, así como sus características, qué ofrecen, su costo para el usuario y las valoraciones de los usuarios finales, para aclarar qué es lo que buscan los usuarios en este tipo de apps y considerar esas necesidades en el software a desarrollar:** en el capítulo 2 se trata el estado del arte, en el cuál se mencionan las apps examinadas y las ventajas/desventajas de cada una respecto al software

desarrollado, por tanto este objetivo se da por cumplido.

2. **Investigar sobre qué herramientas usar para la implementación de la base de datos, backend, frontend, y técnicas y metodologías para dar un desarrollo de calidad al software y para ayudar a la sostenibilidad del sistema en el tiempo:** durante la realización del capítulo de estado del arte se examinaron distintas herramientas a usar, evidentemente no se iban a usar todas las examinadas pero aún sin haber puesto en práctica alguna de las tecnologías examinadas se aprendió sobre su uso y situaciones en las que emplearlas. Sobre las tecnologías previamente ya conocidas, se profundizó más sobre su uso. Este objetivo también se ha cumplido.
3. **Conocer y aprender a usar herramientas actuales y punteras para el desarrollo de aplicaciones móviles:** este apartado está muy relacionado con el anterior. Sin duda lo que más se ha mejorado es el trabajo con la metodología SCRUM, se ha mejorado su entendimiento y refinado conceptos que se creían aprendidos. Cabe resaltar la soltura desarrollada con el framework Express.js(Node.js), que permitió el desarrollo de una API de forma rápida y fácil. Se cumplió este objetivo.
4. **Desarrollar una aplicación multiplataforma que cubra todos los requisitos necesarios para dar soporte a la planificación y realización de entrenamientos y monitorización de rutinas de ejercicio físico:** cabe decir que no se han desarrollado todas las funcionalidades establecidas desde un principio en el backlog, pero se ha cumplido este objetivo, dado que el software da soporte a la planificación con el calendario, a la realización dado que guarda las marcas y ayuda a llevar los descansos del usuario y monitorización porque podemos observar el desarrollo del usuario a lo largo del tiempo mediante las gráficas.
5. **Tratar de obtener un software lo más accesible posible:** No se ha cumplido por completo, dado que algunos aspectos como tener iconos significativos en todos los accesos, se puede hacer complejo dado el tipo de app que es. No obstante, el usuario siempre controla el flujo de la app, el software nunca hace cambios bruscos de pantalla, ni destellos molestos además de poseer una paleta de colores que relaja al usuario.

## 4.2. Valoración personal

Hacer este TFG ha supuesto un gran desafío tanto académico como personal. Me he enfrentado a problemas de desarrollo y diseño reales de aplicaciones completas, como pueden ser fallos en la planificación, decidir si descartar una funcionalidad o no. Lo que más se ha refinado es la forma de trabajar usando metodología ágil SCRUM, que ya se había trabajado previamente en el grado, pero



me ha permitido profundizar aún más, mejorando mi capacidad de organización, priorización de tareas y adaptación a cambios. Además, he tomado conciencia de la importancia del diseño centrado en el usuario y la accesibilidad.

También me ha ayudado a desarrollar mi comunicación, tanto usando esta documentación como ficha técnica como expresando datos en la propia app.

### 4.3. Objetivos de desarrollo sostenible

La integración de principios de desarrollo sostenible en la creación y evolución de la app no solo garantiza su viabilidad a largo plazo, también asegura que el impacto de la aplicación sea positivo tanto para los usuarios como para el entorno. A medida que el mercado y las tecnologías evolucionen, la app puede adaptarse y crecer.

Para ello habrá que centrarse en lo que respecta a la app:

- Optimización para mayor rendimiento y eficiencia energética, se ha tenido en cuenta la optimización de consultas a la base de datos y el uso de algoritmos eficientes para el procesamiento de datos de entrenamientos. La app también gestiona el uso de la memoria interna de forma eficiente,
- Sostenibilidad del ciclo de vida software, para el mantenimiento y actualizaciones a largo plazo. La app está diseñada con un enfoque modular y flexible, lo que permite una fácil adaptación a cambios futuros y a nuevas tecnologías. La metodología SCRUM permite optimizar el desarrollo de una app con un tiempo limitado en términos de calidad.
- Accesibilidad y diseño inclusivo, en esta app se ha tenido en cuenta en la medida de lo posible, siendo compatible con lectores de pantallas y usando diseños que favorecen la inclusión.
- Uso responsable de datos y privacidad. La app cuando pide datos al usuario los asocia a su nombre y no hacia una persona en concreto, además estos datos nunca son subidos a la nube. Por tanto, se cumple esto bastante bien.

### 4.4. Futuro empresarial

Si se deseara convertir esto en un modelo de negocio, habría varias opciones entre ellas:

- Dar una versión gratuita con funcionalidades limitadas y otra versión de pago que de funcionalidades más completas y refinadas
- Se podría limitar el número de elementos en la nube por usuario y si este quisiese ampliar dicho número que pagué por esa ampliación

- Hacer que la app la puedan usar gimnasios permitiendo que los mismos suban rutinas como si fueran un usuario más y ellos serían los que pagan por estar en la nube

## 4.5. Trabajos futuros

En el backlog se han quedado todas las funcionalidades relacionadas con la IA y el smartwatch, las cuales son:

- SCRUM-10** Valorar el entrenamiento en base a la marca actual y la meta del usuario
- SCRUM-11** Monitorizar pulso en tiempo real
- SCRUM-12** Medir pulso en reposo y compararlo con datos de ejercicios
- SCRUM-13** Avisar de anomalías en el pulso de forma suave
- SCRUM-14** Obtener calorías quemadas
- SCRUM-15** Comprobar el equilibrio nervioso del usuario
- SCRUM-17** Conectar con la IA para iniciar diálogo
- SCRUM-19** Resumir datos
- SCRUM-21** Medir SpO2
- SCRUM-22** Interpretar constantes

Todas estas se podrían implementar para versiones futuras. También se podría añadir videos explicativos a los ejercicios, añadir más parámetros de medición a los ejercicios, etc.

## Bibliografía

Leonard, W. P., Mohr, M., & Rønnestad, B. R. (2018). Impact of High-Intensity Interval Training on the Neuromuscular System. *Frontiers in Physiology*, 9, 1367. <https://doi.org/10.3389/fphys.2018.01367>